DOCUMENT RESUME

ED 285 578	IR 051 770

AUTHOR	Dwyer, Frederick D., Ed.
TITLE	Proceedings of the IBM Academic Information Systems University AEP Conference: "Tools for Learning" (2nd, San Diego, California, April 5-8, 1986).
INSTITUTION	International Business Machines Corp., Milford, CT. Academic Information Systems.
PUB DATE	Apr 86
NOTE	679p.; For the conference agenda and abstracts of presentations, see IR 051 771.
PUB TYPE	Collected Works - Conference Proceedings (021) -- Reports - Research/Technical (143)
EDRS PRICE	MF04/PC28 Plus Postage.
DESCRIPTORS	Art Education; Biological Sciences; *Computer Assisted Instruction; *Computer Graphics; *Computer Simulation; *Computer Software; Engineering; Higher Education; Humanities; Microcomputers; Physical Sciences; *Research Projects; Research Universities; Social Sciences; Statistics
IDENTIFIERS	*Advanced Education Projects (IBM)

ABSTRACT

Advanced Education Projects (AEPs) are a series of multidisciplinary, advanced experiments in educational computing supported by IBM and currently in progress at 19 leading universities. Each AEP has established a series of curricular experiments to explore the use of workstations and network technologies as part of the instructional process. The individual papers presented at this conference are grounded in five broad subject areas: (1) Social Science and Statistics; (2) Humanities and the Arts/Design; (3) Physical and Life Sciences; (4) Tools for Learning (software tools, graphics, and subject matter that crosses discipline areas); and (5) Engineering. Graphs, charts, data tables, and references are included for several reports, and indexes by presenter/author and university are provided for each section. A list of program committee members is provided, and an alphabetical listing of all of the authors and their locations is attached. This proceedings volume was originally published containing all 82 reports on IBM's Advanced Education Projects (AEPs) given at the "Tools for Learning" conference. However, 18 papers have copyright restrictions and have not been included in the volume processed (and microfilmed) here. (KM)

****
*   Reproductions supplied by EDRS are the best that can be made   *
*   from the original document.   *
****

IBM

Academic Information
Systems Proceedings

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

☑ This document has been reproduced as received from the person or organization originating it.

☐ Minor changes have been made to improve reproduction quality.

- Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

ED285578

1986
UNIVERSITY
AEP CONFERENCE

April 5 – 8, 1986

Sheraton Harbor Island Hotel

San Diego, California

IR051770

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY
F.D. Dwyer
TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

BEST COPY AVAILABLE

PROCEEDINGS of the 1986
IBM Academic Information Systems
University AEP Conference
"Tools For Learning"

Sheraton Harbor Island East
San Diego, California
April 5–8, 1986

IBM Academic Information Systems
472 Wheelers Farms Road
Milford, Connecticut 06460

John C. Daily, General Manager

Frederick D. Dwyer, Editor
Mary E. Donahue, Associate Editor
William J. Spalter, Technical Director

PREFACE

IBM is sponsoring, at nineteen leading universities, a series of Advanced Education Projects (AEPs). The Projects are multidisciplinary, advanced experiments in educational computing. Each of the universities has established a series of curricular experiments to explore the use of microcomputer workstations and network technologies as an integral part of the instructional process.

The IBM Academic Information Systems University AEP Conference was held at the Sheraton Harbor Island East hotel in San Diego, California from April 5th through April 8th, 1986. In support of the theme "Tools for Learning" the conference focused on presentations concerning, and demonstrations of, AEP experiments.

This publication contains the papers submitted by the conference presenters and is divided into five volumes. Volume I covers social science and statistics, Volume II covers humanities and the arts/design, Volume III covers physical and life sciences, Volume IV covers software tools, graphics and subject matter which goes across the discipline areas, and Volume V covers engineering. The papers are presented without editorial change. Opinions expressed herein are those of the individual authors and do not necessarily represent the opinions of the IBM Corporation.

A number of authors have directed that their materials carry a copyright notice. Papers so marked must not be reproduced without the express permission of the copyright owner.

The typesetting of this publication was done on an IBM RT PC using Interleaf Workstation Publishing Software.

Interleaf and Workstation Publishing Software are trademarks of Interleaf, Inc.

ACKNOWLEDGEMENT

The conference program was assembled, from presentation abstracts, by a volunteer program committee of university AEP managers. They devoted many hours from already busy schedules to the solicitation, collection, review and selection process. Their efforts resulted in a conference widely acclaimed by participants and attendees to be of exceptional value.
We offer a most sincere thank you to:

THE PROGRAM COMMITTEE

Ms. Jacqueline Brown
Coordinator of Distributed Computing Projects
Princeton University

Ms. Cecilia M. Cowles
Project EZRA Manager
Cornell University

Ms. Jean Dexheimer
Assistant Vice President, Computing
Carnegie-Mellon University

Dr. Clifford B. Gillman
Director, Project TROCHOS
University of Wisconsin

Dr. John Goldin
Acting Manager, Project ELI
Yale University

Dr. Glenn Ricart
Director, Computer Science Center
University of Maryland

Dr. Michael S. Sher
Staff Officer for Computing
University of California, Los Angeles

Mr. Chris Swisher
Assistant Director, Computing Resource Center
University of Pennsylvania

Dr. Ronald F. E. Weissman
Director of Instructional Computing Programs
and Director of Project FULCRUM
University of Maryland

PROCEEDINGS OF THE 1986 UNIVERSITY AEP CONFERENCE

Volume I:	Social Sciences and Statistics
Volume II:	Humanities and Arts/Design
Volume III:	Physical and Life Sciences
Volume IV:	Tools for Learning
Volume V:	Engineering

PROCEEDINGS OF THE 1986 UNIVERSITY AEP CONFERENCE

CONTENTS, VOLUMES I - V

Title	Vol	Pg
Computer Assisted Lectures	I	2
Computer Based Instructional Materials in Counseling Psychology	I	13
Enhancing, Analytical Comp'nt, Quantitative Methods: Statistics	I	
Laboratories for Introductory Psychology	I	28
Measuring Speed of Thought: Microcomputer Psychology Laboratory	I	34
Personal Computers, Economics Curriculum, Yale University	I	40
POPGEN: Interactive Simulation Programs, Population Genetics	I	45
Reconstructing A Historical Community: Texas, 1846–1910	I	52
SAMP - A Program to Compute Sample Size	I	62
SPA The Simulated Political Arena	I	
Tools for Learning Data Analysis	I	85
The Great American History Machine	I	97
The Ins and Outs of Macroeconomic Forecasting: A Tutorial	I	108
Treaties, Computers, and Undergraduates	I	
INDEX by Presenter/Author, Volume I	I	119
INDEX by University, Volume I	I	120
A Computerized Sumerian Dictionary of the Third Millennium	II	2
A "Language Lab" for Architectural Design	II	
An Authoring System as a Tool for Foreign Language Learning	II	
Bilingual Word Processing: A New Tool for Learning	II	
Computer Text Analysis/Database Management, Musicological Study	II	45
Interactive Videodisk and Foreign Language Learning	II	

PROCEEDINGS OF THE 1986 UNIVERSITY AEP CONFERENCE

CONTENTS, VOLUMES I – V

Title	Vol	Pg
Musical Acoustics Demonstrations/Lessons	II	60
Philosophical Simulations and Experiments: Scientific Reasoning	II	64
Remote Electronic Music Synthesis Workstations	II	78
Report on Logic: A Workbook	II	
Software for Printing South Asian Scripts	II	91
The Cairo Geniza Project	II	101
Workstations for Computer Music at Carnegie–Mellon	II	109
INDEX by Presenter/Author, Volume II	II	118
INDEX by University, Volume II	II	119
An Intelligent Chemical Stencil for 3D Molecular Coordinates	III	2
Calculus Software From the PC to the RT	III	10
Computer Aided Instruction in Neuroanatomy	III	16
Computer Graphics for an Intermediate–Level Optics Course	III	29
Computer Graphics Utilities in Special Relativity	III	41
Computer Simulations of Plant Morphology and Evolution	III	65
Crystal	III	72
Development of a Computerized Practice Simulation	III	
Dynamic Models in Biochemistry: Using Spreadsheets	III	81
Evaluation of a Bayesian Forecasting Program	III	
Intelligent CIA	III	93
Interactive Graphics/Display & Analysis/Earthquake Seismograms	III	109
IRIS: A Tool for Learning Data Analysis Strategies	III	114

PROCEEDINGS OF THE 1986 UNIVERSITY AEP CONFERENCE

CONTENTS, VOLUMES I - V

Title **Vol** **Pg**

Molecular Structure Graphics for Undergrad Chemistry Ed. .... III 120

New Elementary Physics Laboratory Experiments .... III 124

Representation of Density Distributions on a Sphere III

The Use of Expert Systems to Teach Veterinary Decision–Making .... III 142

Using CAMPS to Teach Medical Problem Solving .... III 147

INDEX by Presenter/Author, Volume III .... III 158

INDEX by University, Volume III.. III 160

A Generalized Videotex Laboratory .... IV 2

A Practical Authoring System/CBI with a Video Digitizer IV

An Administrator's View of AEP Program Operation and Evaluation .... IV 18

Beyond AEP .... IV 23

CMU Tutor:An Integrated Programming Environment for Workstations ..... IV 29

Computer Augmentation of Flight Simulator Training .... IV 38

Computers in Residence: Social and Managerial Issues .... IV 54

Convivial Computing at the Forbes College Computing Center .... IV 60

Designing Local Area Networks for the Academic Environment .... IV 65

Innovations in Higher Education: A Multimedia Device IV

Networking to Support Education Objectives .... IV 76

PDS–A General Authoring System for Development of Simulations .... IV 83

Report on Brown/CMU/MIT Window Manager/Environment Convergence .. IV 92

Scriptwriter II: From Vaporware to Freeware .... IV 98

The Challenge of Qualitative Evaluation .... IV 102

PROCEEDINGS OF THE 1986 UNIVERSITY AEP CONFERENCE

CONTENTS, VOLUMES I – V

Title	Vol	Pg
The IBM PC Network: An Introduction to Managing It Wisely	IV	
Tools for Learning at the Interactive Computer Graphics Lab.	IV	152
Towards an Integrated Design System for the IBM AT	IV	156
You Can Get There From Here	IV	170
INDEX by Presenter / Author, Volume IV	IV	202
INDEX by University, Volume IV	IV	204
A Microcomputer Tool to Design Energy Conserving Buildings	V	
A PC–Based VLSI Design System for Education	V	16
An Instructional Facility for Computer Vision/Image Processing	V	25
An Interactive System Approach to Near Optimal Design	V	36
Closed–Loop Dynamic Testing Control Using an IBM PC/AT	V	57
Computer–Aided Design of Microwave Integrated Circuits	V	73
Computer Graphics Flow Visualization	V	79
Computer Aided System Analysis and Design	V	91
Computerized Instruction in Engineering Thermodynamics	V	95
Development of an Expert System for CAI in Process Control	V	104
Educational Software for Structural Engineering	V	
Interactive Graphics in CAD/CAI in Chemical Engineering	V	
Interactive Microcomputer Graphics Software/Modeling Phenomena	V	168
MCADA: Mechanical Computer Aided Dynamic Analysis	V	179
Teaching Numerical Math Using Interactive Computer Graphics	V	191
The Use of Interactive Thermal Finite Elements	V	196

PROCEEDINGS OF THE 1986 UNIVERSITY AEP CONFERENCE

CONTENTS, VOLUMES I – V

Title	Vol	Pg
VEGA: A Geometric Modelling System	V	
Which Wave Theory	V	215
INDEX by Presenter / Author, Volume V	V	231
INDEX by University, Volume V	V	233

IBM

Academic Information
Systems Proceedings

1986 University AEP Conference

Volume I :

SOCIAL SCIENCE AND STATISTICS

April 5 – 8, 1986
Sheraton Harbor Island Hotel
San Diego, California

CONTENTS, VOLUME I

Title of Presentation **Page**

Computer Assisted Lectures 2

Computer Based Instructional Materials in Counseling Psychology 13

Enhancing, Analytical Comp'nt, Quantitative Methods: Statistics ....

Laboratories for Introductory Psychology 28

Measuring Speed of Thought: Microcomputer Psychology Laboratory 34

Personal Computers, Economics Curriculum, Yale University 40

POPGEN: Interactive Simulation Programs, Population Genetics 45

Reconstructing A Historical Community: Texas, 1846–1910 52

SAMP – A Program to Compute Sample Size 62

SPA The Simulated Political Arena ....

Tools for Learning Data Analysis 85

The Great American History Machine 97

The Ins and Outs of Macroeconomic Forecasting: A Tutorial 108

Treaties, Computers, and Undergraduates ....

INDEX by Presenter/Author, Volume I 119

INDEX by University, Volume I 120

COMPUTER ASSISTED LECTURES

Presenter: Steven M. Goldman *University of California, Berkeley*

DESCRIPTION:

Economic theory most commonly is taught with the extensive use of diagrammatic representations of formal mathematical models. Indeed, lecture presentations frequently rely almost exclusively on such graphical techniques to characterize economic relationships. The capacity, then, to display the graphs for a particular model (and to view the consequences of modifications in the underlying parameters) affords a powerful expositional device for the discussion of a wide range of problems in both micro and macro analysis encompassing such techniques as:

1. comparative static and dynamic analysis,
2. the existence and determination of partial and general equilibria, and
3. the displays of optimization techniques.

This project is concerned with the development of programs to support the presentation of lecture material in economic theory courses. These programs are then to be available for the subsequent use of the students; enabling them to review and experiment with the lecture material.

TECHNICAL CONSIDERATIONS:

When the models in question are essentially quantitative — such as representing historical data — then the graphics power of existing "spreadsheet" type programs provides a convenient tool for the presentation of this material. Often, in the presentation of theoretical material however, it is not the "data" which is interesting but rather the display of a functional form or a surface which is of interest.

The programs themselves may be viewed as either specific — oriented toward the presentation of a narrow group of concepts — or general — utilities designed to assist in the display of mathematical relationships which would support a variety of topics.

For the development of specific routines, it is essential to hold the programming time to a minimum since each program constitutes at most a single class lecture. In order to facilitate the rapid development of such material, the programming medium must be rapid to use and easy to debug. The speed of final code is rarely the primary consideration. STSC's implementation of APL was selected for this function and performed successfully. The friendliness of an interpreted language and the excellent graphics support, along with the ability to create a library of frequently used "functions", eventually cut the development time for lectures to a matter of hours.

There are however, two major disadvantages associated with this use of APL. Foremost is the current inability to produce direct machine code. Thus, all users of the program require the appropriate Interpreter -- an expensive piece of software. Secondly, if the software is to support student interaction, familiarity with APL notation is required. Often students who were unfamiliar with computers in the first place, found their first experiences the more mysterious by the appearance of APL code. Nonetheless, the convenience and power of the language as a tool for rapid development is impressive.

The development of general routines offers several potential advantages in reducing development time. If common features of the various lectures are sufficiently pronounced, then it becomes practical to create a diagram generating program to serve as a draftsman. The repeated use of such programs make their optimization somewhat more attractive. To redress the complaints with APL - principally the requirement of the interpreter for final users - Turbo Pascal was chosen as a substitute for the next stage of development. The author's experience has been most satisfactory w.r.t. speed and flexibility. The major disadvantages experienced thus far have centered about the failure to explicitly support the IBM EGA and the notorious 64K limitation on the Code and Data segments.

The former concerns were fortuitously overcome by frequent excursions into PC Tech Magazine and CompUServe from which it was possible to assemble a substantial library of graphics routines. The 64K limitations proved less fearsome than anticipated. The ability to use the pointers and to place data in the "heap" along with the capacity to create overlays reduced this concern to a relatively minor inconvenience.

EXPERIENCE:

The specific programs described below cover a variety of topics in Economic Theory spanning both upper–division undergraduate and graduate courses.

Lecture programs have been written covering:
1. General Equilibrium in an Exchange Economy
2. General Equilibrium and Production
3. Fair allocations
4. Scitovsky contours
5. Duality Theory in Production
6. Two Sector General Equilibrium
7. Optimal Economic Growth
8. A simple Hicks trade cycle model.
9. Derivation of Indifference Curves through Revealed Preference
10. Construction of Average and Marginal Cost Curves
11. The Solow One–Sector Neoclassical Growth Model
12. Construction of Von Neumann–Morgenstern Utility Functions
13. General Equilibrium and the Brower Fixed Point Theorem.
14. Utility Maximization subject to constraint
15. Hicks Income and Substitution Effects

Other programs currently under development include:
1. IS–LM analysis
2. Multiplier analysis
3. Production Externalities and Efficiency
4. The Theory of Monopoly
5. Cournot Oligopoly Theory
6. Imperfect Competition.

One lecture was given in a graduate theory course illustrating the use of the calculus of variations in the solution of a "turnpike theorem". A single PC was used to provide output to some 20 monitors dispersed about the classroom. The ability to respond visually and accurately to questions pertaining to changes in the parameters of the model was most effective. Since such facilities are not yet generally available, most of the remaining programs could be used only in a tutorial setting.

These programs are described in greater detail in the Appendix below.

The development of a general utility program by this project has only recently been undertaken. Thus far, it enables the interactive display and manipulation of functions of a single variable, i.e. $y = f(x)$, projections of functions of two variables, i.e. $z = f(x,y)$, and data sets with three arguments. These three dimensional figures, whether functional or data, may be conveniently rotated by the user to gain different views of the underlying relationships. The program is menu driven and a series of context sensitive help menus are available. Graphs may be saved, restored displayed in windows and manipulated on the screen. This device has applications to building an understanding of the basics of many two dimensional projections which pervade economics as well as allowing a more sophisticated visualization of higher dimensional relationships. These techniques have proved useful in Elementary Theory courses for illustrating the concept of level sets, in Econometrics for generating displays of bivariate distributions and Likelihood Functions, in Mathematics for Economists by illustrating general principles relating to maxima, minima and saddle points and in Game Theory for examining the nature of a complicated payoff function. Other applications are being developed that are directed toward the presentation of specific economic material.

(See Figure 1.)

APPENDIX:

The following programs have been written under the STSC APL/PLUS for the IBM PC and have been used in the teaching of Economic Theory at the graduate level.

CONDOT -- This is an experimental (read still some bugs) program which allows the specification of an arbitrary function $z=f(x,y)$ along with a list of levels for z. The contour maps for each of these levels are then displayed. The program frequently runs into APL's limitations on workspace and object size. (The idea for this program comes from BYTE magazine and is there implemented in BASIC).

COSTFN -- Starting with an arbitrary cost function, sets are constructed depicting those combinations of factors which cost no more that the minimum cost for a variety of factor prices and the specified level of output. Since the isoquant can not intersect any of these sets -- or else the minimum cost would be less than the value of the cost function -- the resulting union of these sets must form a lower bound to the isoquant at the chosen level of output.

DEMAND -- This program has numerous variations. Essentially, you are asked to specify the parameters for a CES function and a list a price and income combinations. The resulting diagram(s) show the utility maximizing choices for the price income combinations selected. If the price of only one of the goods is varying across the price income combinations submitted, then a right diagram returns the corresponding demand curve. Similarly, if income alone is varying, the right diagram shows the Engle curve. The program may also be called in "conversational" mode where you will be prompted to input the parameters as you proceed.

DUALITY -- Proceeds from the specification of the parameters for a CES production function to find the cost minimizing factor bundle for a range of input prices. The left diagram shows the factor space -- initially drawing the unit isoquant and then the tangent with the specified factor prices. The input prices are then normalized to produce a cost of 1 and the right diagram depicts -- in factor price space -- the combinations of input prices which would also cost exactly 1 using the cost minimizing factor bundle on the left. This process is repeated for a collection of factor prices. The final right diagram depicts the convex hull for the unit isocost curve.

(See Figure 2.)

EDGE -- This program allows for the specification of parameters for two individuals' CES utility functions along with their endowments. First the Edgeworth box is shown with some representative indifference curves. Next you may act as the auctioneer, suggesting various ratios of exchange between the two commodities and the resulting choices by the agents are shown. This may be repeated. Finally, you may select the competitive equilibrium and the offer curves,equilibrium price and choices will be displayed.

FIRM -- First, the average and marginal costs are derived (on the right) from a total cost curve (on the left). Then, you may specify an inverse demand function (i.e. price as a function of quantity) and the left graph will display the total revenue curve while the right will illustrate the average and marginal revenue curves.

(See Figure 3.)

FIXED -- This routine illustrates the fixed point mapping for identifying a competitive equilibrium in an exchange economy with three goods. In one form, a representative collection of starting prices in the price simplex are taken and their direction of "movement" is shown. Alternatively, an arbitrary starting set of prices can be taken and, treating the map as a procedure for selecting the next price vector, the stable approach to equilibrium can be demonstrated.

HDEMAND -- This program illustrates the Hicks-Slutsky decomposition for a Cobb-Douglas type utility function. You are asked to specify the parameters for the utility function along with income and initial and final price for good 1. The points marked O,M,H,S are the (O)riginal demand bundle, the (M)arshallian demand bundle after the price change, the (H)icks bundle where income has been adjusted to keep the same level of utility and the (S)lutsky bundle where income is adjusted to just enable the purchase of the original bundle (O).

PROFIT -- A simple production process is considered where the first 'A' units of labor produce no output but are required as input before production can take place. A variety of price to wage ratios are then used to maximize profit. The resulting hull contains the production set but is unable, of course, to identify input-output combinations which would never be utilized -- i.e. those generating negative profits since the firm would "shut down".

REVEAL -- An illustration in two-space of the approximation of an indifference curve using the strong axiom of revealed preference. First, a budget set and chosen point are displayed. Next, all of those points which could be directly revealed preferred to that point are shown (i.e. those above the price consumption curve through that point). The "indifference curve" must pass through the zone of ignorance - the space between the budget set and below the price-consumption curve. Then another point on the budget set is taken and prices and income are internally computed so that this point would be chosen. It's budget set is then displayed. Bundles in the union of the two budget sets are then indirectly revealed inferior to the original point. This procedure is repeated numerous times with the union of all such budget sets being indirectly revealed inferior to the original point. Finally, the "true" indifference curve is shown for comparison.

SOLOW -- This routine provides for a dynamic illustration of the Solow one sector growth model. The diagrammatic long-run equilibrium is displayed and the convergence w.r.t. time is illustrated in a second window.

UNCERT -- You are asked to respond to a number of lotteries offering either a certain payoff or a gamble between two possible outcomes. You supply the probabilities for the gamble for which you are indifferent between the certain payoff or the gamble. Finally, your "utility" function is displayed along with some implications from a consistency check. If you were indeed consistent, then these consistency points should lie along the curve. The idea for this procedure comes from Richard Gilbert.

* The author has made liberal use of the ideas and even the code of others in the creation of these programs. Some of Professor Mark Rubinstein's APL routines have been absorbed as subroutines. An excellent set of EGA routines in Turbo-Pascal by Kent Cedola have been downloaded from CompuServe and are used for the color graphics in the Turbo-Pascal Program. Specifically, other routines appearing in the PC Tech Journal on the Programming of the EGA, on High Resolution Printing and on hidden line removal have proved extremely valuable.

References:

Bridger, Mark and Mark Goresky, "High-Resolution Printer Graphics", BYTE, November 1985, pp. 219 - 232.

Cedola, Cedola, "TPEGA" -- Turbo Pascal routines for color graphics with the EGA. (CompUServe).

Chandler, Richard and Gary Faulkner, "The Painter's Algorithm", PC Tech Journal Vol. 3, No. 11, November 1985, pp 181-187.

Dean, James L., "Calc.Pas" --An expression Parser in Turbo Pascal. (CompUServe).

Giloi, Wolfgang K., "Interactive Computer Graphics", Prentice-Hall: New Jersey, 1978.

Louderback, Joseph G., "Rapid Screens in Turbo", PC Tech Journal Vol. 3., No. 9., September 1985, p. 39.

Mansfield, Victor., "Scientific Graphics with the EGA", PC Tech Journal, Vol.3, No. 9, September 1985, pp 163–170.

Rubinstein, Mark E., — Various utility subroutines in APL.

Steiner, David D., "Taking Command in Turbo Pascal", PC Tech Journal Vol. 4, No. 3., March 1986, pp161–177.

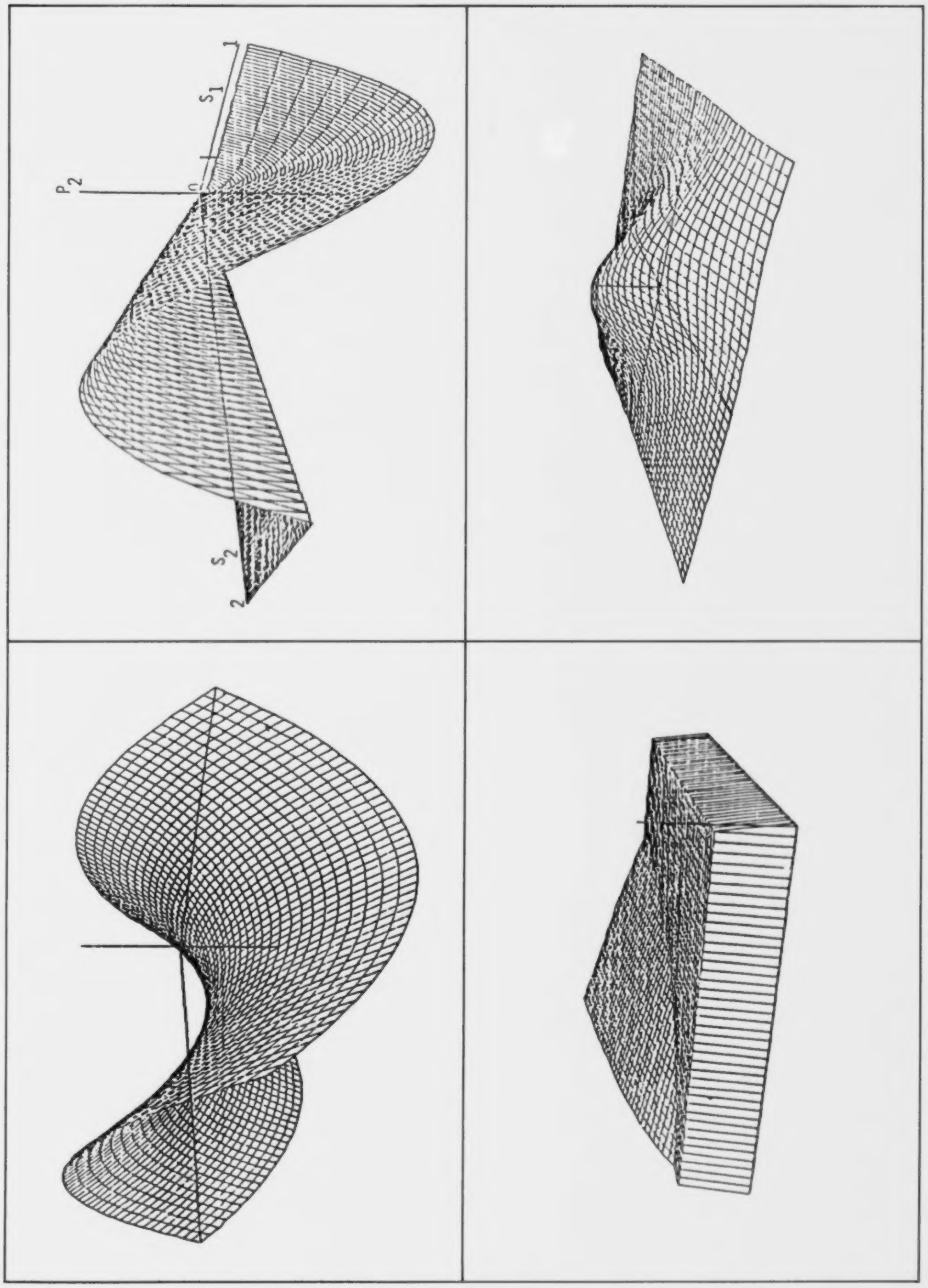


Figure 1.

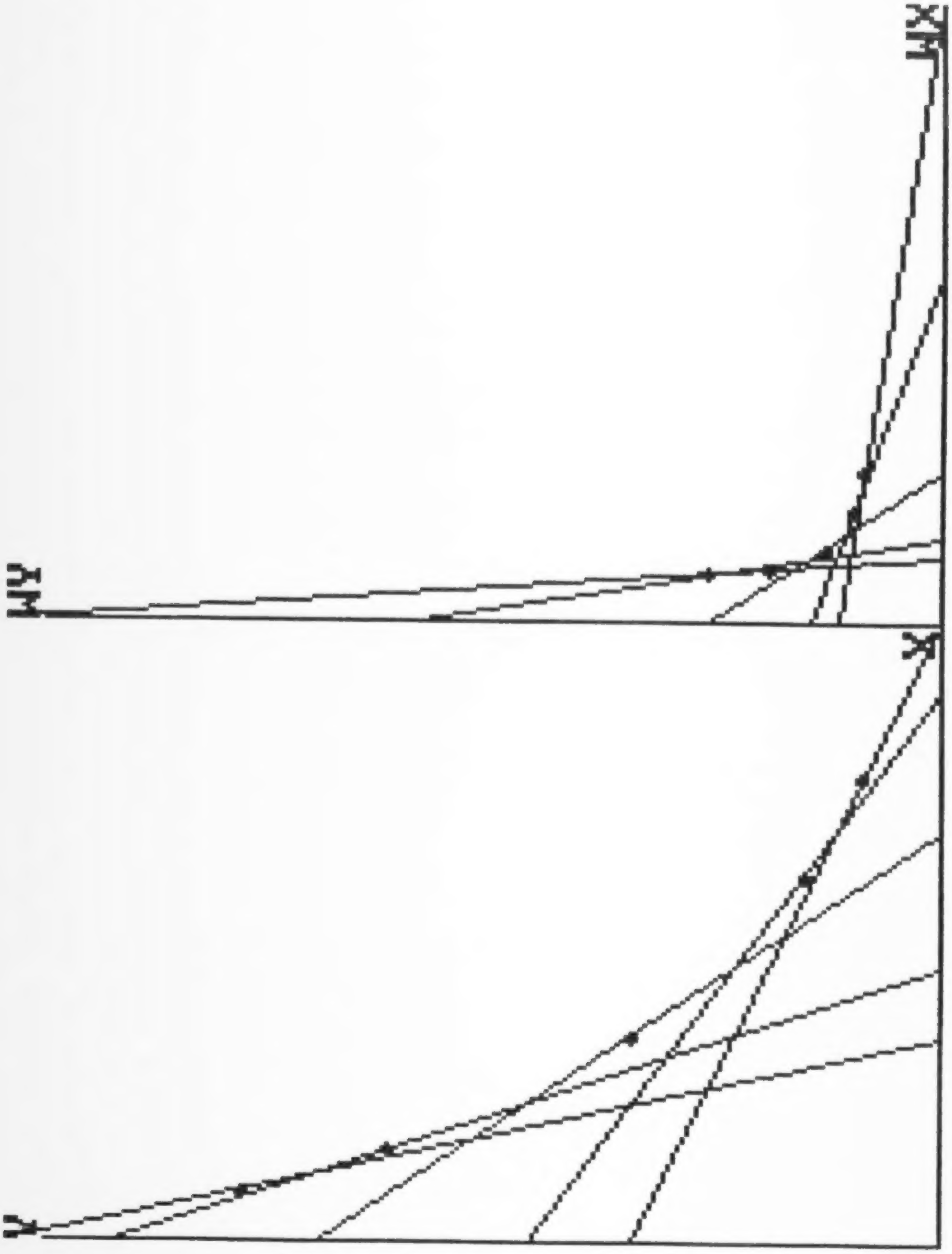


Figure 2.

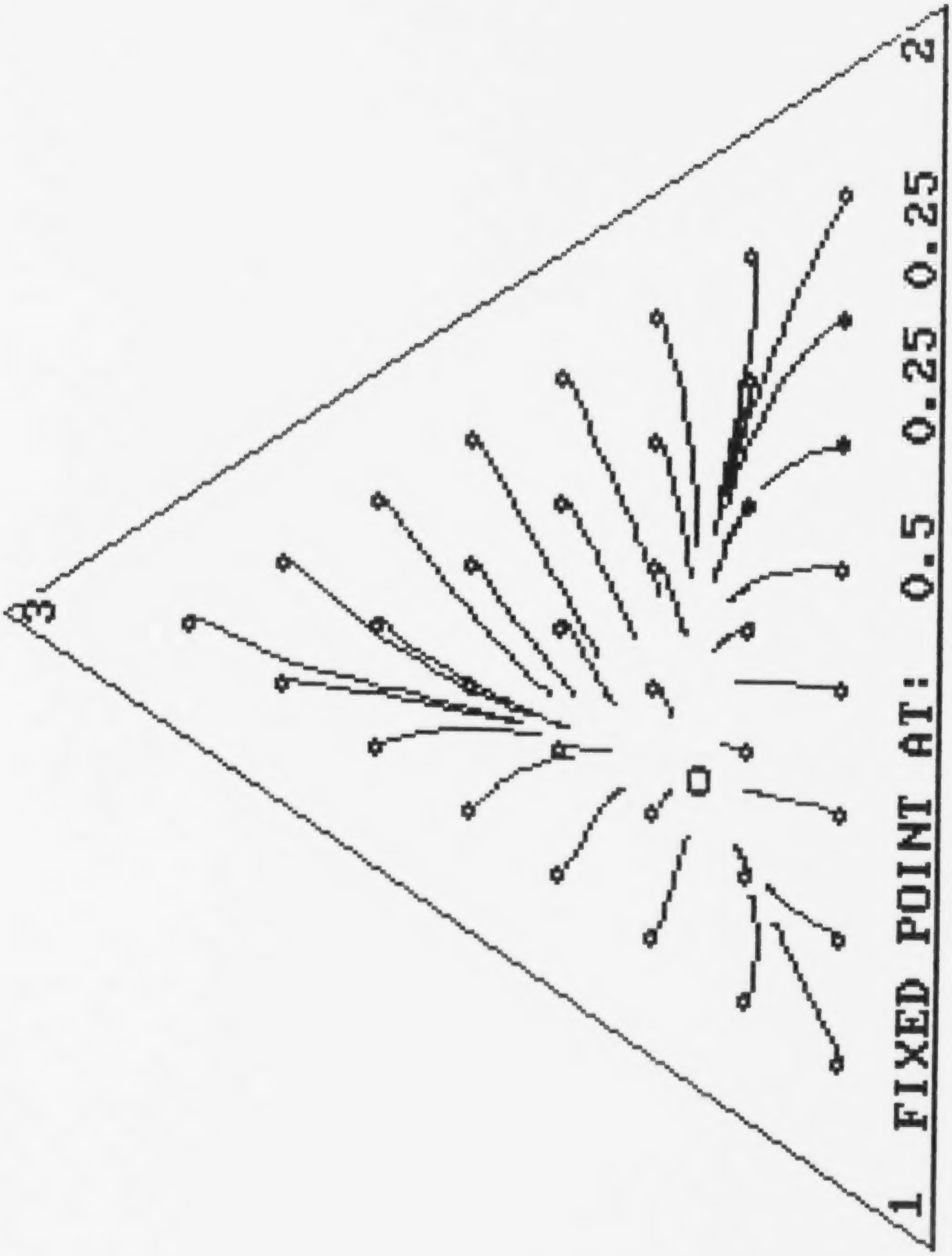


Figure 3.

COMPUTER BASED INSTRUCTIONAL MATERIALS IN COUNSELING PSYCHOLOGY

Presenter: James L. Lee *University of Wisconsin, Madison*

copyright © James L. Lee, 1986

BACKGROUND:

The department is a graduate studies program for master and doctoral students preparing to work in schools, universities and a variety of public/private agencies. The department does offer several undergraduate service courses for non-majors: Theory and Practice of Interviewing and three one credit modules dealing with self-development as a studnet/person. The self-development modules focus on Educational Effectiveness (study skills), Career Development, and self-Awareness. Finally, the department is also involved in direct client service delivery through our counseling practica and internships. The department has its own small clinic with audio/video observation/recording capabilities.

Our AEP project is focused on four educational goal areas.

Educational goal 1: Increased student awareness and development: personal development, study skills, and career exploration.

The purpose of our three undergraduate self-development modules is to enhance individual student growth around the three areas mentioned. One of the major educational processes normally used in these types of courses is individual testing with commercially developed and scored instruments. The department usually enrolls 120 to 150 students each semester in these self-exploration modules. University policy prohibits the direct charging of students for commercial testing processes as part of a course. In addition, each of the one credit modules lasts for only five weeks. The usual time lag involved in commercial scoring makes it difficult to use commercial instruments. Our purpose was to develop an exploratory environment in which students could assess various aspects of their development or behaviors around the three topical areas of self, study skills and career exploration. Each of the modules is taught in small groups of 12 to 18 students by a Teaching Assistant/Counselor with an emphasis on both small group interaction and individual counseling.

We have developed a three disk series, each containing four self–exploration inventories directed at the topics of the modules. The first task was to identify topical areas that are normally taught in each of the modules. The second task was to design the twelve inventories in paper and pencil form and initially assess their reliability and validity. Since the purpose of the inventories is self–knowledge and personal growth to be used in small group discussion and individual counseling, our emphasis was on content validity rather than predictive or concurrent validity. The inventories were designed to give students immediate feedback concerning the meaning of their scores. There were two other considerations in the planning. The first was student confidentiality. Students should be allowed to respond to the inventories and receive feedback anonymously. At the same time, there should be the possibility to save student scores and item responses for retrieval and use by the Teaching Assistant/Counselors.

The programming development of the disks was quite straight–forward. Because of our previous experience, the programming was done in BASIC. Each disk and inventory program follows the same format.

A. Introductory Menu: The introduction was written to emulate a typewriter as it provides an explanation and presents the menu options. Initially the menu included the four inventories and a file manager. The file manager incorporates a password system so that only the TA/Counselors can review individual student scores and responses. We found that, even with a password system, students would try to access the file manager and would destroy and/or access other student data. We had to remove the file manager from the main menu and make it accessible to the TA/Counselors through a hidden key function.

B. Each inventory asks for student name and permission to save the scores on the disk. If the student allows, his or her scores and item responses are saved to disk after they have completed the inventory and received feedback. If the student does not allow, he or she may still complete the inventory and receive feedback.

C. The feedback the students receive can be obtained either on the screen or through printed copy for further reference.

D. The file manager allows the TA/Counselor to obtain a list of all students who have completed the inventory and review an individual's item responses and scale scores. This allows the TA/Counselor to develop counseling interventions specific to each student. The file manager also allows for some limited analysis: item mean scores and standard deviations for males, females or the entire group of respondents. This allows the TA/Counselor to develop group interventions.

The three disks and inventories are as follows.

A. Self-Exploration Series 1: Who Am I?

1. Money Values
2. Parent Relationship Scale
3. Stress Inventory
4. Self-Concept

B. Self-Exploration Series 2: Study Skills

1. Study Attitude Quotient
2. Learning Styles Inventory
3. How do you Study?
4. How do you Think?

C. Self-Exploration Series 3: Career Exploration. Since there are a number of excellent job/career search and matching programs on the market, this disk focuses on some of the underlying issues and concepts necessary for making career decisions.

1. Orientation Inventory: Internal vs. External
2. Occupational Matrix: People vs. Things
3. Occupational Values
4. Occupational Planning: A career maturity planning index that teaches basic concepts.

It has been difficult to assess any improvements in the educational experience of the students since we have introduced a new process. Because of the financial and time constraints of commercial inventories, we have never used such materials in the courses. The response of the students, however, has been extremely positive. The TA/Counselors report that students are eager to use the software and the TA/Counselors are better able to identify individual trouble areas and develop more appropriate interventions/teaching materials.

Educational Goal 2: Counseling students will learn the elements of accountability/effectiveness and will learn to state counseling goals in behavioral and measurable terms.

A major issue in our field precipitated by declining social service and school budgets is counseling effectiveness and accountability. Counselors/psychologists, in any setting, need to be accountable to their profession (adding to the body of knowledge through research), need to be accountable to administrators, and need to be accountable to their clients. Counselors/psychologists are accountable for their use of resources, for their time, and for achieving counseling goals (both programmatic and individual). Two specialized data based systems have been developed to help our counseling practicum and internship students account for their utilization of agency resources and to assess their counseling effectiveness. Through the use of these data bases counseling psychology students are introduced to the major components of accountability: time and goal attainment. In addition, the use of the data bases and the automatic analyses frees the students to devote more time on underlying principles and theoretical concerns rather than on record keeping and statistical data manipulation. These educational goals could have been met by teaching students data base concepts and how to utilize existing data base management systems, such as dBase III. However, the students and/or professionals involved are relatively unsophisticated in computer use, data base concepts, and software utilization. It was reasoned that we could develop a data base system which would incorporate all of the basic concepts and statistical analysis normally used without having to teach students underlying data base concepts. Such a system needed to have some flexibility so that it could be adapted to a variety of settings. In this regard we developed two such data base systems: The Counselor Log and The Counseling Goals System.

The Counselor Log: Using the recommendations of the State of Wisconsin Department of Public Instruction, we developed a data base system to allow counselor/psychologists in training to be accountable for their time. Students in practica and internships keep data concerning: Type of Person met with; Type of Activity engaged in with that person; Outcome of the meeting; Source of Referral; Grade Level of the person; and Time spent. Each of these categories provides the user with a listing of possibilities to choose from. The users type in the number corresponding to their choice, e.g., the type of person. Data is saved by date and can be entered from a normal appointment calendar each day or periodically. While the elements in each category are those recommended by the Department of Public Instruction, The Counselor Log allows each user to customize all of the elements within each category. The customizing capability allows each student to design the Log to meet the unique needs of individual settings. Data analysis provides a number of standard statistics: Total time spent with what kind of person doing what kind of activity; Total time spent in each activity at grade level; Total number of contacts by person with average time spent; Total number of contacts spent in each activity with average time spent; Total number of referrals by source; Total time and number of contacts at each grade level with average time spent; and Total number of contacts by defined outcome levels. Data can be analyzed by day, week, month, semester – to a full year. Data can also be combined across any number of counselors/psychologists as long as they are each using the same definitions. The Log provides us a way of seeing how students actually spend their time with some measure of effectiveness – outcomes. When data is combined across counselors/psychologists, there is the basis for programmatic evaluation.

The Counseling Goals System: Assessing counseling effectiveness has always been problematic. Each counseling client and situation is unique. The problem occurs when attempting to compile data across a number of unique cases. It is metaphorical to comparing apples with oranges with grapefruit, etc. The second problem is to teach students to systematically define counseling goals in ways that are measurable. A system called The Minnesota Goal Attainment scaling System (GAS) was developed under federal grants which addresses these issues. The GAS system is in the public domain and was used as the basis for developing the second database system for counselors/psychologists. Students keep client records as follows: Client name; Client age; and Client sex. The student then enters up to three goals for each client record. Each goal must further be behaviorally defined at three levels: Worst possible outcome expected on each goal; Expected level of achievement on each goal; and Best possible outcome expected on each goal. Since the record can consist of up to three goals, and since all the goals may not have equal importance, the student then weights the goals by assigning a one or two digit number 0–99) indicating the relative importance of each goal for each case. The student keeps track of the number of counseling sessions used with each case and the type of counseling intervention used for the case. There is additional space to allow the student to keep a limited amount of notes concerning the case. When the counseling case is closed the student indicates the level of goal outcome achieved by the client on each goal. The level of outcome and the goal weights are used to compute a standard goal attainment score for the case. The scores range from 33–66. A score of 50 indicates the goals for the case were achieved at the Expected level. Standard scores 50 or above indicate the student counselor helped the client achieve the counseling goals at the expected or above expected level of achievement. Standard scores below 50 indicate the counseling was below the expected level. The standard scores can then be combined across cases to analyze counseling outcome effectiveness: accountability to administrators and clients. The analysis procedures built into the system allow the student/counselors to assess their effectiveness by most all the combinations of case record elements. The Counseling Goals System will automatically compute means, standard eviations, ANOVA and t–Tests using research questions developed by the student. For example:

a. Which intervention(s) was(were) most effective on particular counseling goals?

b. Which goal(s) was(were) successful using a particular intervention?

c. Which goal(s) was(were) most successful by sex, grade/age?

d. And so on. There are twenty possible analysis procedures range from simple lists to 11 different ANOVA and t–Test analyses.

The Counselor Log and Counseling Goals System are combined on one disk with a password security system to maintain confidentiality of the counselor/psychologist records. The programs were written in BASIC and, even though not compiled at this time, operate very fast. The final product is essentially the same as our original design with one major exception. The analysis portion of The Counseling Goals System required that goals or interventions be typed in exactly as they were originally entered so that data could be compiled appropriately. This added a special burden or a two step process for the user: list all the goals or interventions and obtain a printed copy so that they could be entered accurately and completely during the analysis section. The latest version has been developed so that data can be searched or analyzed by key word(s). This has simplified both the use and analysis process. Each masters student is required to complete a research seminar paper as part of the program. Practica and internship students using this system have been able to complete their papers in a more timely manner. Because of the demands of practica and internship settings students often delayed graduating as much as a semester to complete their research papers. Since the data analyses are done "automatically" for students, they have been able to spend more time dealing with the theoretical issues involved in their research rather than the unproductive time of data statistical manipulation. Faculty whose students have used the system report that the quality of the research seminar papers has improved substantially. At present, the student is required to enter two sets of data–Log and Counseling Goals. Future refinements planned are to combine both systems into one data base. The primary lessons learned during this part of the project relate to user friendliness. We are dealing with both students and professionals who view the computer as antithetical to the nature of the profession. We found that very simple assumptions for people who have any computer familiarity could not be made. For example, the length of an allowable input could not be controlled in the usual manner of giving sound and visual feedback when the input length was too long. This was frustrating to the users because they would then have to retype the entire line. This necessitated the writing of a customized input routine which automatically controls the length of the input, i.e., stops when the maximum limit is reached, but still allowed for full editing or use of the appropriate function keys. We have found that computer applications to our field, beyond the usual wordprocessing or data base system, meets with great resistance.

Educational Goal Area 3: Provide direct counseling client services through the use of computer technology.

We are examining ways of using the computer to deliver counseling services. In this regard we have developed a weight loss/eating habits computer enhanced training program which includes the use of subliminal suggestion. The program consists of an eight week system, combining the use of written materials with the computer to teach effective eating and weight loss concepts; to set and monitor weekly goals; to monitor eating behaviors; and to provide subliminal suggestion around typical eating problem areas. After an initial introduction and weight loss goal the system is designed to be used three times a week. Each week follows a similar format:

A. Mondays: Client weighs in and enters current weight and then is shown a weight loss chart to date accompanied with appropriate feedback. Client then reads written lesson and answers comprehension questions on the computer. Client reviews eating behaviors for last two days – receives points toward his or her reward. If the client achieves the requisite points, he or she is told they can reward themselves with the reward previously chosen. Client enters a new reward for the next week. Client enters eating and behavior goals for the next two days. Finally, client is asked to visualize three problem eating scenes and watch screen for subliminal suggestion.

B. Wednesdays: Client reviews eating behaviors and eating goals for last two days – receives points toward reward based on responses. Client sets new eating behavior goals for next two days and finishes with visualizations and subliminal suggestions.

C. Fridays: The process is exactly the same as Wednesdays with a visual representation of the points earned toward the established reward.

The weight loss system was developed in BASIC. This system has been used in two different doctoral research projects. In both cases the experimental groups, those receiving subliminal message and those who thought they were receiving subliminal messages, lost an average of more than 10 pounds in a seven or eight week period compared to no loss for an inactive control group. In the first study an additonal 4 week follow up was done to check continued post–experimental session effects, in the experimental group that received actual subliminal suggestion, 6 of 9 subjects continued to lose or maintain weight loss. In the experimental group that received nonsense subliminal suggestion, 6 of the 9 subjects gained weight at the second follow up. The soft ware seems to be effective as a direct counseling service. Based on the suggestions and feedback from clients who participated in the program, there will be additional changes made in the programs before additional use and research: the addition of a "counselor" for the client to interact with; more personalized feedback and monitoring relative to individual eating patterns, etc.

Educational Goals Area 4: Students will learn to properly identify and respond appropriately to linguistically ill-formed client verbalizations.

The last area of focus is presently under development and is the greatest challenge. A great deal of counselor training revolves around learning to discriminate and respond appropriately to client statements, statements which are usually called linguistically ill-formed. We are developing response skill training software which will eventually be combined with video cassette or videodisc.

Linguistically ill-formed refers to client statements that place responsibility on others. There are an identifiable number of these statements, e.g., Implied causative: "I want to leave home but my father is sick." The implied causative is indicated by the "but" and the implication that father being sick is somehow preventing the client from leaving home, thus, placing responsibility on father's sickness. Counseling students need to learn to first identify such ill-formed statements and then learn to respond to them in a man ner which places responsibility back on the client. The software is in the design stages at this time.

We are designing an expert system which includes several functions. First, the user is presented with a client statement and asked to identify the kind of ill-formed category with appropriate feedback and available tutorial. Second, the user will be asked to provide an appropriate counseling response. The user's response will then be evaluated and feedback will be given indirectly. That is, if the response is judged to be appropriate, the continued responses from the client and counseling interchange will proceed in an effective, predictable direction. If the user response is judged to be inappropriate, the client responses will veer in unproductive and unrelated areas. Finally, users will be given feedback as to their performance in identifying and responding appropriately with suggestions for additional practice. Eventually, we want to combine this process with video cassette or videodisc to allow the user to take advantage of the subtle nuances of nonverbal behavior, voice volume, tonality, and speech rate.

Initial attempts to develop this software in BASIC have proven too difficult and unwieldy. Our present efforts are directed at using one of the artificial intelligence languages such as Prologue or LISP. It is hoped that we can then develop a prototype expert system which can be used by our faculty to teach a variety of response skill modes.

LABORATORIES FOR INTRODUCTORY PSYCHOLOGY

Presenter: David R. Williams *University of Pennsylvania*

I will describe my use of the PC as a kind of animated textbook, addressed to a general undergraduate audience. I will focus on the considerations that went into the construction of its first "chapter", on behaviorism. My presentation will necessarily abbreviate the experience of working through the chapter as a whole: students normally spend 2 hours, or a bit more. When we go through the Laboratory today, I will skip most of the textual material, and focus primarily on the interactive displays. These--more precisely, the active process of learning they encourage--are the heart of the Laboratory.

The purpose of the Laboratory, and the way the PC helps accomplish it, derives from the special teaching challenge posed by Introductory Psychology--and I gather that many of my colleagues in other Psychology Departments face a similar situation. Introductory Psychology serves a broad spectrum of students: from those for whom the course will stand as the required, and thus exemplary, course in science, to those hard-science concentrators for whom Introductory Psychology is the deepest penetration yet in a humanistic direction, voluntary or otherwise. It is common to have students of both types in the same lecture hall. What does one say to each? To all? There is a mix of years and genders, representing a diversity of experience, a clash of veneers of sophistication, and a multiplicity of special hopes for the course. For some, this is an introduction to a Major program of study; for others, a "terminal" course in the field. How does one make the course valuable to them all?

These questions would be rhetorical if they were not practical, and even urgent, in light of the high enrollments Introductory Psychology sustains in many Universities. They are answered, in practice, every time a section of Introductory Psychology is taught. The series of Laboratories for Introductory Psychology that we are developing is addressed, experimentally but non-rhetorically, to this problem. In general, we try to use the capabilities of the PC to provide examples of phenomena discussed in text and lecture, and to permit students to experiment with selected variables. The phenomena we choose--the illusions, for example--are intuitively compelling and easily produced with a PC. Like classics in the arts, such phenomena appeal in many ways, and are thus useful for a broad audience.

The particular Laboratory I will demonstrate today does not use the PC in the usual experimenter–subject format; rather, under instructions from the student, it carries out a simulation of operant behavior controlled by several sources of reinforcement. The components of an overall simulation model are introduced one by one, and their internal parameters are explored. Then the components are combined, and one of the combination rules is made available for exploration as well. The component models are intuitively stated and computationally simple. They focus attention on the relation between the parameters of the model and the graphs generated by their use. Exploration of the parameters is guided by the intuitive, everyday terms in which the model is cast: all parameters refer to variables that have dual citizenship, with correlates in both behavioral and experiential domains. The connection between quantitative process in the model and the experiential referent is emphasized by the sights and sounds of the plotting process, whenever a new run of the model is carried out.

A final comment on the relation of the Laboratory to Introductory Psychology. I believe the Laboratory makes 3 important but quite different contributions to the educational impact of the course; all are valuable, and influenced program design decisions. That these particular 3 issues came together here is perhaps accidental: the only common property I can find is that together, they seem relevant and interesting to a broad plurality of our students, and tend somehow to co–occur whenever I teach the course.

1. The Behavioristic Perspective is a struggle to teach, and it is difficult to avoid seeming absurdist or simplistic. By "Behavioristic Perspective" I mean the insistence that concepts and theories be based on information derived from observable actions or action–patterns of an organism, under specified circumstances. A bit rigorous, perhaps, but this concept is at the root of the natural–science psychology. I mean "behavior" strictly, but in a way that classifies both signal–detection analyses and all instances of inter– personal communication as having a significant behavioral component. This astringent approach to what we can study in scientific psychology is both foreign and unwelcome to our students. It is worth persisting with because, beyond its historic place and the weight of its empirical utility, such ostensive behaviorism is an important part of scientific discipline in many of the current clinical

applications of psychology and their extensions to psychiatry; not to mention the relatively ignoble strictures of consumer–behaviorism, where theory is replaced by the bottom line of spending activity, nor the more honorable behavioristic distinction between "observed" and "inferred" so common to legal thinking.

2. To think behavioristically is also to think abstractly in terms of, say, the probabilities of your response measures, under circumstances defined by an act–outcome relationship. Analyzing reinforced operant behavior is like describing an ecology of action, an economy of environment/behavior relationships conforming to general laws. So to think behavioristically means to think quantitatively; and this again is very foreign to many Introductory students, even when put more concretely than this. Can the PC help here?

3. The final contribution deals with neither process nor substance, but with the enterprise as a whole. Real behaviorism is a disciplined point of view; and however great its value, every point of view has cost, because it emphasizes some things at the expense of others. One cost of behaviorism is a lack of relevance to "inner experience": the private world of the stuff of conscious life. How bad a price is that? It didn't cost much to think of the colors of the planets as "secondary qualities", to be ignored in favor of their mass, a "primary quality" of scientific importance. Reasonable enough for planets, but what about psychology? Does the behaviorist account leave out anything important? That is a prototype question appropriate to all natural–science psychology. If the PC can help a student gain, however briefly, the behaviorist perspective, it can help a student answer this most important of questions––important, at least, as a matter of liberal arts education.

With all this background as a statement of the problem to which our Laboratory is addressed, let me describe the overall model and then demonstrate its central features. The model considers the "studying behavior" of a "Typical Penn Freshman", as a function of conflicting sources of reinforcement. Each source is itself governed by a different quantitative model, whose parameters the student can set via "options" screens. The models are chosen to provide reasonable depictions of the intuitive flow of events in TPF's life during a typical semester. To help clarify differences in parameters, and aid in differentiating their effects, auditory feedback accompanies visual plotting of the graphs.

The student enters a large laboratory room with 12 PC stations. Each is equipped with a joystick and a color monitor. Under simplest circumstances, the student turns the machine on (do so), or follows instructions on a general screen. By either route, the following welcome screen appears: its purpose is to establish rapport and suggest an active approach to the experience.

Even under the most optimistic of circumstances, this and every other Laboratory of the Introductory series will need continuing revision in light of ongoing experience. That fact was taken into account in the program design. The screen of text that you see now comes from a file of full-screen images stored on diskette; in addition to the image, each record of the file also carries an additional 16 bytes that are used for programming: they control the color of the image, but more important, can determine the "next" and "last" text pages and procedures. This information is carried by the records themselves, so that they effectively control program flow between screens. This makes modification very simple, and has greatly aided revision of the Laboratory.

The records that carry the text and sequence information are prepared by a separate editor program. Because the records carry sequence codes, they can be inserted and withdrawn into chains,included in subchains, etc. Because text and options screens are all stored in an external file, changes do not require recompiling the Laboratory program.

The next page shows the student what lies ahead. This screen is a recent development, and is intended to give an overview of the Laboratory as a whole. Let me go over the screen with you briefly.

Now we will go to the options screen for the Exam-Driven Study model. The student is confronted with an "options menu", and can choose options by moving the joystick around. We have found excellent acceptance. The numbers are clear and easy to change, the operation of the stick becomes natural very quickly, and there is little opportunity for error. Many of our students are initially afraid of touching a computer, but have adapted very quickly to the joystick.

The "run" screen shows these parameters, plotted for a fictitious semester, as described in the pages I skipped over.

The "hints" screen shows the default values, and tells how to explore the model. The text screens have explained about studying as an avoidance response, gradients of reinforcement, and the rules of the simple model that generated this curve. The gradients are just decay curves, the parameters are actual percent fall–off rates, and there is a lower bound.

The student may exit to the Random Distraction model at any time. The intervening text supplies a background, introducing the concepts of choice, response competition, and individual differences.

Next comes the "Social Life Model". Conceptually, it is a markov process with an anomalous state, exit from which is a function of time rather than trials. The purpose of this anomalous state is to permit modeling of the "opponent–process" theory of motivation. This theory provides a realistic means of simulating effects that are at least subjectively real and memorably felt, thus enlisting intuitive support. It also provides a means of introducing the final "individual difference parameter" into the simulation, thereby providing a demonstration of how important one's style of handling frustrative non– reward turns out to be.

Let me leaf quickly through: this text explains the model first intuitively, and then in terms of stages. We arrive at the options screen: let me demonstrate and explain.....

The model can only be run in isolation from this screen. There is one more step to add as we progress to the fourth stage. As the next screen shows, when we combine this model with the others, we must decide how the Inhibitory Units affect other responding. They could plausibly either enhance the effectiveness of regressive reinforcers like pillow–cuddling, or they could energize an avoidance response like Exam–Driven Studying, in line with the maxim of the "...tough get going". How much difference this makes to the overall semester depends on the generalization gradient, also a student–controlled parameter. Thus the combined model permits considerable experimentation. Throughout, it is intended to encourage the development of intuitive quantitative thinking, bridging the gap between abstract model and computer–driven representation. Such encouragement, accompanied by practice at maintaining a consistent behavioral frame of reference in the extended analysis of a situation, is the major contribution of the PC.

Let me show you the displays....There is one more feature: the summary display. (Show). This is intended, again, to provide practice at relating different quantitative representations of the same empirical process, and to foster an appreciation for the merits of different quantitative descriptive formats. (Show).

The system has evolved considerably since I first tried it on a very small scale. It is still in a tinkering stage, and introduced in a low-key manner. Well over 100 students have tried the program so far: a group self-selected, of course, for their interest. Their comments have been favorable and encouraging on the whole, and nearly all describe the program as "friendly" and "easy to use". The joystick and the sound effects are both popular.

I have no way to be determine, yet, what effect this laboratory program will have on the effectiveness of the Introductory Course. The first trial semester is still in progress, and change is still rapid. Evaluation of the effectiveness of this and the other Laboratories is an important step once the current experimental phase is complete. At present, I gather my feedback informally, and use it to develop the program, not to validate it. Meanwhile, and as a necessary condition always, this and similar Laboratories can only fill their potential if they inherently sustain interest and involvement. We do not, at present, require participation in the Laboratories, although we recommend it. The program does seem to support sustained involvement by the students, who are participating voluntarily, and it is also interesting for me to develop and update–– easy too, because the screens are simple to prepare and insert in the programmed series, and are capable of controlling the instructional flow. It seems to me that the PC makes an interesting new teaching medium available to us. I have enjoyed working with it; I hope your brief excursion has been interesting.

MEASURING THE SPEED OF THOUGHT: A MICROCOMPUTER BASED PSYCHOLOGY LABORATORY

Presenter: Kevin F. Miller *University of Texas at Austin*

Many of the most fundamental psychological processes happen so quickly that we have little awareness of how they occur. Reading a word or finding the larger of two numbers, for example, occur in roughly the time it takes to blink an eye. Because of the speed with which these cognitive events transpire, we have poor intuitions about the processes involved. Yet accurate measurement of the time required to perform basic tasks such as recognizing words reveals a great deal of structure even in processes that require only about a second to complete.

The speed and inaccessibility of basic cognitive processes create two related problems for instruction and research. The speed of such processes means that they their description requires precise timing and control of stimulus displays, while their inaccessibility makes it difficult to convince students of the reality of hypothesized processes underlying word recognition or other basic cognitive processing.

To meet both instructional and research needs for demonstrating and studying the processes underlying rapid processing, we have developed an IBM PC-AT based system that permits accurate measurement of such brief processes along with immediate display of results. This system is well-suited both for classroom demonstrations of basic cognitive processes and for research applications. The remainder of the paper will describe design considerations in the development of this system, its implementation on the IBM PC and PC-AT, the phenomena demonstrated at this conference, and our research directions.

DESIGN CONSIDERATIONS FOR A COGNITIVE PSYCHOLOGY MICROCOMPUTER LABORATORY

Three distinct classes of users will utilize this system. For classroom applications, users must be assumed to have very little sophistication in operating computers. Therefore, in a demonstration mode, the system must be essentially "bullet-proof", requiring a minimum of unfamiliar responses on the part of users. A second class of users are graduate and undergraduate students who are interested in designing their own studies. While a greater level of computer sophistication can be assumed for these users, the system must provide them with a tool box of basic routines, which can be selected from and modified for a wide range of experiments. A final class users will be children of various ages, who will interact with the system as subjects. Of course, little if any computer facility can be assumed on the part of these users. Therefore development of simplified response modes (speech recognition and simplified response boxes) has been incorporated into the system.

HARDWARE ISSUES

The IBM PC-AT as a laboratory microcomputer. Studying brief cognitive processes requires the ability to rapidly present high-quality stimuli, collect responses from a variety of modalities, and accurately measure the time (reaction time between stimulus display and response.) The PC-AT is well suited for our purposes.

1. High quality displays. The characters typically displayed on microcomputers are not really adequate for research purposes. Fortunately, the Enhanced Graphics Adapter (with additional graphics memory) permits one to create one's own fonts in a variety of sizes. We developed a rudimentary font editor that permits one to design larger and more detailed characters, yielding a vastly improved stimulus display.

2. Speech and switch input. We need to be able to measure a variety of responses, including the time required to speak words or to push buttons. The Voice Communications package permits us to use spoken responses. We are able to find out when someone began to speak a word (which we use for timing purposes), and the system also implements speech recognition of isolated words in a flexible and sophisticated way. We are able either to provide a general template (which we use for this demonstration), to train a new vocabulary for each speaker, or to adapt a prior template to the pronunciation of a new speaker. The speech recognition system is particularly useful for computer naive users, and it can greatly expand the range of responses permitted within a general response-time paradigm. Switch input is accomplished through the parallel port. The standard parallel port provides a way of reading switches such as buttons, permitting accurate polling of the time at which a button was pushed.

3. Timing. We require the ability to time intervals to a one-millisecond resolution (.001 seconds). The AT (and the IBM PC) have hardware clocks that are ideally suited for these purposes. In the relevant mode, the hardware clock works by repeatedly decrementing a preset value, generating an interrupt upon reaching zero. By changing this divisor value, one can generate interrupts at any desired frequency, including the 1-millisecond level of accuracy that is standard in cognitive research. By writing a separate interrupt service routine, one can keep track of the interrupts and provide the operating system with interrupts at the ordinary rate. Thus, the timing modification is transparent to other aspects of the computer that normally make use of the hardware clock.

SOFTWARE ISSUES

1. Building a toolbox of routines. The system is designed to be modular, consisting of a number of functions that can be rearranged to meet the demands of a variety of experiments with minimal reprogramming costs. Considerations of speed,modularity, and the need for interfaces with assembly language programs led us to write the system in IBM/Microsoft "C".

2. Graphic displays of results. For demonstration purposes, immediate displays of results are critical. The Plotting System provides a fast device-independent means of producing a variety of graphs. For two of the demonstrations here, results are written to a file which is then used as input by a plotting routine that immediately shows individual results. A simple change in the calling command can redirect the graph from the display to a plotter, printer, or other graphics device.

DEMONSTRATIONS AND INSTRUCTIONAL APPLICATIONS

Self-running programs have been developed that illustrate three classic phenomena in cognitive psychology. They demonstrate the existence of automatic or obligatory psychological processing (Stroop, 1935), illustrate processes used in word recognition (Meyer & Schaneveldt, 1971), and show the existence of a very brief "iconic" memory (Sperling, 1960).

1. Automatic processes. If some cognitive process can occur automatically, or without conscious attention, one is free to deploy attention elsewhere. In reading, for example, the fact that as adults we don't have to think about what words are in front of us frees us to think about higher-level questions (e.g., "Is

this making sense?" or "What's he trying to pull on me?"). Poor readers often are forced to pay attention simply to recognizing the words, making it difficult for them to attend to the meaning of passage (Just & Carpenter, 1980).

A side effect of automatic word recognition is that it is very difficult for a skilled reader not to read a word. Stroop (1935) demonstrated this in a elegant way, showing how difficult it is to perform a simple task (naming colors of ink) when some of the stimuli were color names. Thus, it takes longer to decide that the word "RED" written in green ink is green than it does for some string of letters written in green. The interference shown in this task shows a negative consequence of what is generally a very valuable process – the automatic recognition of words.

2. Lexical decisions. Deciding whether or not something is a word is a task we can do very quickly. The decision is affected by a number of factors, two of which are 1) how "word-like" a pseudo-word is, and 2) how frequently a word occurs in English text. This task requires you to press one of two buttons to indicate whether or not a string of five letters is a word. The stimuli consist of pseudo-words, some of which are homophones of real words (e.g., "rore"), and words, which vary in their frequency of occurrence (Kucera and Francis, 1967). Even for highly-skilled adults, deciding whether or not a string is a word is affected by both factors – familiarity of the word, and the rules that determine how word-like a string is.

3. Information available in brief visual presentations. The final task shows the existence of a very brief memory for unanalyzed visual information. We are able to keep images around for a fraction of a second before we recognize them. The task demonstrating this will show you a 3 x 4 matrix of letters for a brief interval (100 milliseconds), which is then cleared. After another interval a low, middle, or high tone will sound and you should try to recall the letters in the row corresponding to the tone. The ability to recall a row drops drastically in the interval from 50 to 500 milliseconds, as this "iconic" information decays.

RESEARCH APPLICATIONS

Research applications of the system have focused on studying developmental changes in the processes underlying number processing. Prior studies (Ashcraft & Battaglia, 1978; Miller, Perlmutter, & Keating, 1981) suggest that there is a developmental transition in calculation from rapid counting-based procedures, to a table–lookup process of calculating in adults. We are collecting data on the time required to perform simple arithmetic in conditions of mixed vs. isolated operations, as well as processes used in rapid counting and number naming.

Overall, the system has proved to be flexible in research applications, and both motivating and convincing to students in psychology courses. Measurement of the speed of rapid processes provides a way of making accessible important but otherwise obscure cognitive processes, and microcomputers have a critical role to play in making an understanding of these processes available to ordinary undergraduate students.

REFERENCES

Ashcraft, M. H., & Battaglia, J. (1978). Cognitive arithmetic: Evidence for retrieval and decision processes in mental addition. Journal of Experimental Psychology: Human Learning and Memory, 4, 527–538.

Just, M. A., & Carpenter, P. A. (1980). A theory of reading: From eye fixations to comprehension. Psychological Review, 87, 329–354.

Kucera, H., & Francis, W. N. (1967). Computational analysis of present–day American English. Providence, RI: Brown University Press.

Meyer, D. E., & Schaneveldt, R. W. (1971). Facilitation in recognizing pairs of words: Evidence of a dependence between retrieval operations. Journal of Experimental Psychology, 90, 227–234.

Miller, K. F., Perlmutter, M., & Keating, D. (1984). Cognitive arithmetic: Comparison of operations. Journal of Experimental Psychology: Learning Memory and Cognition, 10, 46–60.

Sperling, G. (1960). The information available in brief visual presentations. Psychological Monographs, 74, No. 11.

Stroop, J. R. (1935). Studies of interference in serial verbal reactions. Journal of Experimental Psychology, 18, 643–662.

PERSONAL COMPUTERS IN THE ECONOMICS CURRICULUM AT YALE UNIVERSITY

Presenter: Matthew D. Shapiro *Yale University*

Although the general public certainly perceives economics as an experimental science, economists' experiments are carried out not in the controlled environment of a laboratory, but in the uncontrolled environment of the world economy. Although many economists attempt to persuade the Congress, the Federal Reserve, and the President to carry out our experiments on the witting or unwitting public, the lack of laboratories puts economists at a distinct disadvantage relative to natural scientists in both testing our theories and teaching students the fundamentals of our discipline. The Congress, the Federal Reserve, and the President are unlikely to turn over levers of monetary and fiscal policy to Yale freshmen in order that they may learn experimentally the consequence of policy actions on economic outcomes. (I have, nonetheless, sometimes thought the Yale freshmen might not out–perform the policy authorities.)

In this paper, I describe how we are using personal computers in our curriculum to bridge the gap between textbook models and practical applications. In a wide range of courses and at levels from introductory economics through graduate econometrics, we are using exercises on personal computers as laboratories for conveying to our students what we hope is a more fundamental understanding of economic theory than that obtainable through conventional instructional methods. Moreover, as the gap between the number crunching ability of personal computers and mainframes narrow, personal computers are being used much more in economic research. Hence, our use in courses of personal computers introduces our students specializing in economics to an important tool they will be using throughout their careers.

I will discuss applications of PC's in teaching macroeconomics and econometrics that are now underway in Yale's economics department.

MACROECONOMICS

Economists have, for the last thirty years, used large scale macroeconomic models for forecasting and for evaluating the consequences of economic policy actions. These models can be thought of as mathematical embodiments of standard theories of the behavior of economic aggregates. Although they are much more complex and detailed than the algebraic and graphical representations of the theories presented in introductory texts, they essentially trace out the same interactions but with richer dynamics and more detail about the different sectors of the economy.

These models have long been estimated and simulated on mainframe computers. Because of the complexity of the models, and the difficulty of using mainframes, these implementations are largely the province of specialists and certainly not very useful as teaching tools. Recently, such models have become available for PCs. For example, Fair's model is available on PC.(1)

One of the most difficult, yet important, points in economic analysis is simultaneity. Use of the macroeconomic simulation models allows students to trace out for themselves these effects and come up with explanations of the various components of the the changes.

To be concrete, consider the analysis of a simple but important policy change: the Gramm–Rudman deficit reduction. In analyzing this policy in class, we might outline the following chain of causation. Output falls immediately as consequences of the spending cut. There are multiplier effects: for a dollar of reduction in government spending, output fall by more than a dollar because consumers cut their spending. Moreover, because income falls, tax receipts falls, so it will take more than a dollar in cuts to cut the deficit by a dollar. Finally, and most importantly, what happens to income is controlled by the Federal Reserve's reaction to Gramm–Rudman. Only if it fails to loosen monetary policy when fiscal policy tightens will output ultimately fall.

This scenario is simple enough to ask a freshmen to analyze at the computer. She will be given the basic facts––the path of the deficit and the possibilities for monetary policy. She will be first asked to design expenditure cuts to achieve the deficit reductions. This is complicated by the feedback from the expenditure cuts to tax receipts through the reduction in income. She will then be asked to analyze the short run and long run effects of the policy on output, interest rates, and inflation under different monetary policies and to study how important each channel is quantitatively.

Variables such as government expenditure enter the model exogenously. Money can be made to either react or to maintain an exogenous path. Once the model is solved, the student is shown the consequences of the experiments in terms of the absolute levels and the change relative to some base simulation. The results can also be plotted. The task is then to "come up with a story" for the numbers generated. A student might focus on say, the consequences of Gramm–Rudman for the interest rate. The interest rate will fall modestly as a direct consequence of the deficit reduction, but it will fall dramatically if Fed tries to counteract its contractionary consequences. Most of the lessons of introductory macroeconomics can be illustrated in terms of these policy interventions and how the economy reacts to them.

In addition to coming up with explanations of how the reactions take place, the student sees the quantitative consequences of the interventions. For example, simple textbook models imply that a decrease in the deficit will permanently lower output. Later in the course, students are told how the economy will eventually return to full employment even if the deficits is permanently reduced and no other policy comes into play. In the model, she can see how long this takes (about 2–3 years) and by what channels (lower interest rates and prices).

We are using the macroeconomic models at two levels in the undergraduate curriculum. First, two sections of introductory macroeconomics use them for exercises such as I have described. Second, a seminar of seniors use the models to do much more detailed analysis including their own modification and enhancement of the models. The seminar students report their results in term papers or senior theses.

ECONOMETRICS

Teaching econometrics is the second major area where we are currently using PCs in our courses. Two areas are Monte Carlo simulation and estimation.

In undergraduate statistics, we use real time, graphics–oriented Monte Carlo simulations to teach the Weak Law of Large Numbers and the Central Limit Theorem. Students use programs that calculate and graph the distribution of normalized sums of independent observations. They can choose parameters such as the number of observations and the standard deviation on the underlying random variable. They then see graphically how fast (a) the means of these sample converges to a spike, and (b) the distribution approaches normal. Seeing is believing. We find that these experiments lead the students to a concrete and intuitive understanding of these basic statistical theorems.

In econometrics, we use the GAUSS(2) programming language for student exercises in the graduate course. GAUSS is a matrix, APL–like programming language available only on PCs. It was developed specifically for econometrics. At the graduate level, econometrics is largely taught in terms of matrix algebra. Students attend lectures that involve complicated matrix algebraic derivations. They also do these themselves in pen–and– pencil homework exercises and examinations.

We also want to give the students some practical experience in estimation with real data. In most econometrics courses, students estimate regressions using canned packages on mainframe computers. In these packages, the formulas giving the estimators are coded in FORTRAN, which the student never sees. They estimate regression models by using "canned" procedure. Hence, the gap between the analytic work and the paper–and– pencil exercise is huge. Although the exercises do serve to introduce the students to evaluation of actual, empirical results, they fail to solidify their understanding of the basic analysis.

GAUSS has bridged the gap between the paper and pencil and the computer. In GAUSS, the student writes formulas in almost exactly the same symbols as are used elsewhere in econometrics. For example, the familiar least squares formula

$$b = (X'X)-1X'y$$

can be written in GAUSS as

$$b = inv(X'X)*X'y \; .$$

Students get numerical results not out of some black box, but as the straightforward application of what they have learned elsewhere. In GAUSS, conceptual mistakes, such as inverting a singular matrix, are immediately apparent. Moreover, students can experiment with small matrices to verify that properties which are derived analytically actually do hold in the appropriately transformed data. If they find their transformed data do not have the correct properties, they know that their implementation or perhaps their understanding is inadequate.

Finally, we teach a number of techniques in class that are not easily done in canned packages but are straightforward to do in GAUSS. Asking students to do something that is not routine is an excellent way to get them to work step by step through a derivation.

We have some weak evidence that use of GAUSS has improved the performance of the students in their analytic, paper and pencil work. On last year's midterm, before using GAUSS, about one-third of the students made the serious error of multiplying non-conformable matrices. This year, no one made the error. We believe this is in part a consequence of their manipulation in GAUSS on the numerical matrices.

In order to gauge students reactions to using the PCs for their exercises, we conducted a survey at the end of the semester.(3) Half the students reported experimenting with the software beyond what was required to carry out the assigned exercises. All but one student reported that the computer exercises were at least somewhat helpful with the analytic work; twenty-five percent reported that they were very helpful.

SUMMARY

We are very pleased with the progress we and our students are making in narrowing the gap between conceptualized experiments and actual ones. As more faculty see these benefits, we expect our use of PCs in teaching will grow rapidly.

FOOTNOTES

1 Ray C. Fair is Professor of Economics at Yale University. The Fair Model is available for the PC from Economica, Inc. See Ray C. Fair, "Macroeconomics in the Classroom: No Longer Just Theory," Perspectives in Computing 5 (Summer 1985) 11-19, for further details.

2 GAUSS is a product of Applied Technical Systems.

3 The class was made up of eighteen graduate students, mostly from the economics department. There were sixteen responses to the survey.

POPGEN: A PACKAGE OF INTERACTIVE SIMULATION PROGRAMS FOR TEACHING POPULATION GENETICS

Presenter: **Kenneth K. Kidd** *Yale University*

Coauthors: **Kenneth K. Kidd** *Yale University*
Randy D. Holmes *Yale University*

Population genetics attempts to describe the forces that cause changes in genetic systems, i.e. evolution, and to understand how those forces operate and interact in causing genetic change. Many of the factors involved––natural selection, random genetic drift, linkage, etc.––can be described by simple mathematical equations relating the gene frequencies in one generation to gene frequencies in the subsequent generation as functions of parameters representing the relevant forces. Because of these natural mathematical relationships, population genetics is the most mathematical, and in many ways the most abstract, of the sub–areas of genetics. This presents special problems in teaching this subject to biology students, a sizeable percentage of whom eschew mathematics. While it is possible to emphasize the mathematical and formal theoretical aspects of population genetics, it is also possible to emphasize the consequences of the various relationships rather than the mathematics itself. It is in this area that interactive computing can be most helpful.

We have written a program package, called POPGEN, which incorporates the mathematical relationships and, interactively with the student, will simulate changes in genetic systems from one generation to the next. The POPGEN programs have been written in PASCAL using the Turbo PASCAL compiler. The choice of PASCAL was motivated partly by the recursive nature of many of the calculations. The current version of the program requires the 8087 math coprocessor and color graphics. The main program requires 26 K bytes of memory; the overlay modules total another 100 K bytes, but no more than half of that is in memory at any one time. The core mathematical routines currently represent a miniscule fraction of the source code. The majority of the code is devoted to interactive I/O to allow the student to enter the values of the parameters and check on their accuracy so that the simulation done will be exactly the one the student wishes. Numerous checks for incompatible parameter values and for values out of range have been built into the program to prevent crashes of the program that might result from arithmetic errors possible with nonsensical values, e.g., division by zero.

While virtually all population genetics textbooks present graphs of specific examples of evolutionary change, those textbook examples cannot convey to a student the range of possibilities or allow the student to perceive the generalities present over wide ranges of parameter values. Students could carry out the necessary recursive calculations by hand (most are algebraically simple), but such calculations are tedious, often very time-consuming (especially if carried out 100–500 times in succession), and highly prone to error. The personal computer is an ideal instrument for facilitating such experimentation on the part of the student. The student is able to choose the specific parameter values of interest to her or him and almost immediately see the consequences of those values for a particular population. The fast response is a critical factor in promoting experimentation. A personal computer with graphics capability is the ideal machine for such programs because the results are most readily perceived by the student when presented in graphic form. The results of interest, such as gene frequency, are usually between fixed limits (between 0 and 1 in the case of allele frequency) allowing a simple graphical representation to be easily incorporated into the program. In fact, extensive use of color graphics has been the most challenging programming problem. Options exist within the POPGEN programs to allow the student to choose the most useful representation for each problem. Hard copy on a graphics printer can be generated for selected results so that students can have permanent records.

Historically, the package of programs grew from the simple initial design of computer programs that would carry out the iterative numeric calculations involved in projecting the future evolution of a population under a certain genetic model with certain starting values for the relevant parameters. The package has developed along three lines. First, the interactive prompting and error checking of values has been elaborated so that the program prompts the student for all of the relevant values, gives the student options to change values already entered, and checks to make sure that the values entered are within the biologically and mathematically relevant range (e.g., a gene frequency must have a value no less than 0 and no greater than 1). Second, the display of the results has been converted into color graphics rather than numeric output. Some numeric precision is lost but precise numerical values are of secondary consequence; directions of change through time, patterns of change, and rates of change are the didactically important results of the simulations. Third, an interactive tutorial component has been added at several points where reasonably simple algebraic calculations can predict the ultimate outcome of a given simulation. In such cases, the student is asked for her or his prediction and the student's response compared with the theoretical calculation. The student is given the option of observing "help" screens giving the relevant formulae and a chance to modify the initial guestimate. The student is then told whether the last answer is correct, numerically close but not quite correct, or sufficiently far off to be considered wrong. The "close but not quite correct" response was programmed in to encourage the student to do rough mental calculations without being denigrated by the program for not having the precisely correct answer. Even the "correct" response only requires accuracy to two significant figures.

The POPGEN package currently consists of five basically different simulation programs, several of which have options that introduce additional variety. Each program begins with a flow chart of the simulation and, at the student's option, one to two screens of text describing the simulation and giving relevant background.

1. The simplest program to run simulates random genetic drift––the change in gene frequency from generation to generation that will happen just because of sampling error in a small finite population. The only relevant variables are the starting allele frequency and the size of the population. Each generational cycle the program simulates binomial sampling of the gametes to produce the next generation from a pool of gametes that has the frequencies of the genes in the current generation. This follows quite closely the logic of the simulation of random drift discussed by Cavalli–Sforza and Bodmer (1971).

2. The program to simulate selection at a two–allele autosomal locus is seen by the student in three different versions. The simplest simulates deterministic selection given a starting frequency and relative fitnesses for the three possible genotypes. The other two versions introduced inbreeding and stochastic factors resulting from finite population size.

3. The program for selection at an X-linked locus with two alleles allows the student to choose whether or not stochastic factors will be considered in addition to the student's specification of the five relevant fitnesses—three in females and two in males. When the student eliminates both selection and stochastic variation but starts with different gene frequencies in males and females, this program generates the "classic" graph seen in all textbooks: an oscillating asymptotic approach to equal gene frequencies in the two sexes (see, e.g., Crow and Kimura, 1970; Hedrick, 1983).

4. The program to simulate deterministic selection at an autosomal locus with three alleles requires that six fitnesses be specified by the student in addition to starting gene frequencies. This program uses a different display format for the results: the changes in gene frequencies through time are plotted in a de Finetti diagram (see Li, 1976). Such a triangular coordinate system is most appropriate since it allows any set of three allele frequencies to be represented by one point in the graph.

5. The final program simulates the much more complex model of two loci that are linked. Not only must the student choose the fitnesses for all ten two-locus genotypes but must specify starting gamete frequencies and a recombination frequency. The output of this simulation consists of graphs of both allele frequency changes and gamete frequency changes through time. Figure 1 shows the hard copy output a student can get for the gamete frequency plot and illustrates the complex patterns that can be simulated. It is precisely the complexity of evolutionary changes in a population from such two-locus models that is one of the more important concepts that can be so readily conveyed to students using this program.

Many other population genetic models can be simulated in similar ways and it is anticipated that several more programs will be added to the POPGEN package over the next year or two.

Initial tests on selected individuals indicate that the didactic purpose of these programs is well met. Individuals find the programs fun as well as very illuminating. The first full classroom test of these programs is underway in the Spring 1986 semester. We hope the students will find the programs easy to use and sufficiently fun that they will not only ask the "what if...?" questions but will find out the answers for themselves. Use of these programs should give students an intuitive feeling for the relative importance of the several different factors that affect evolution.

ACKNOWLEDGEMENTS

We want to thank Mr. Matt Hawley and Dr. Andrew Pakstis for their advice and help during the writing and debugging of these programs.

REFERENCES

Cavalli–Sforza, L.L. and W.F. Bodmer, 1971. The Genetics of Human Populations. San Francisco:Freeman Press. 965 pp.

Crow, J.F. and M. Kimura, 1970. An Introduction to Population Genetics Theory. New York: Harper and Row. 591 pp.

Hedrick, P.W., 1983. Genetics of Populations. Boston:Science Books International. 629 pp.

Li, C.C., 1976. First Course in Population Genetics. Pacific Grove:Boxwood Press. 631 pp.

Figure Caption

Figure 1. Changes in gamete frequencies over 400 generations.

In this graphic result from simulation of a two–locus model the frequencies of the four types of gametes, listed in corresponding order and "color" on the left, are represented by the vertical widths of the bands. The population starts with AB at very high frequency. The other three gamete types all increase in frequency initially, but later two of them, ab and Ab, decrease in frequency. Eventually an equilibrium is reached with all four gamete types remaining in the population.

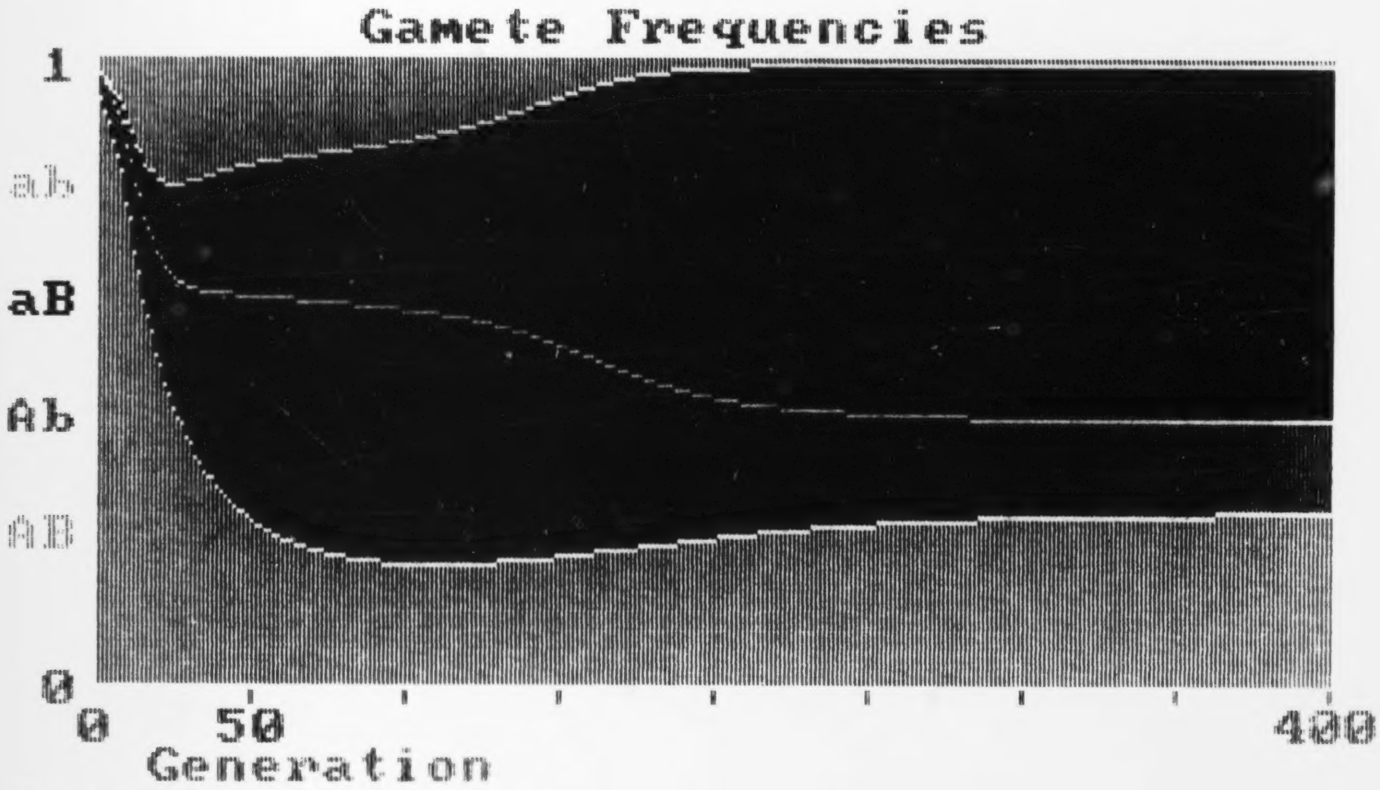


Figure 1.

RECONSTRUCTING A HISTORICAL COMMUNITY: FREDRICKSBURG, TEXAS, 1846–1910

Presenter: **Myron P. Gutmann** *University of Texas at Austin*

Coauthors: **Myron P. Gutmann** *University of Texas at Austin*
Kenneth H. Fliess *University of Texas at Austin*

1. EDUCATIONAL AND COMPUTING GOALS

The study of history involves the amassing and connecting of many pieces of evidence which all pertain to the same person, or place, or event. Historians collect information about something, which they sort, and study, and sort again, in order to make sense of it. Thus, the history of a great war, although it has many episodes and many participants, involves the collection and management of information in order to understand the hows and whys, just as does explaining the hows and whys of the history of an individual person who lived his or her relatively uneventful life within a single town, holding only a few jobs, and participating in only that community's events.(1) If the life history of a war involves millions (if not billions) of individual data, the history of our person involves tens or perhaps one hundred. The point we are making here is that there is a universality to the collection and management of information by historians, a universality which differs only by the scale at which it operates.

The obvious corollary is that data management of this type is perfectly suited to the computer with its quick and capacious memory, its ability to tirelessly sort and resort information, and its capacity to churn through its files repeatedly without losing anything, without making dog-eared its notes, and without getting crabby in the way most graduate student spouses who were put to these tasks in the past did. Computers are ideally suited to all sorts of tasks in historical research, but they are for nothing better suited than thecollection and organization of large quantities of life histories, of the sort collected by historical and anthropological demographers.(2)

There is nothing novel about what we are reporting here. This is old knowledge to professional historians and advanced graduate students. Many of us have used computers extensively for twenty years now and an increasing number have wholeheartedly (if clumsily) joined the movement toward personal computers.

What is not going nearly so well is the transfer of this knowledge to our undergraduates in liberal arts colleges. If you ask students in the liberal arts program at the University of Texas about microcomputer use, you discover that the most serious academic work to which computers are being put is word processing. The reasons are simple: we lack courses which use computers in the social sciences and humanities; we lack faculty capable of teaching computers for these subjects; and we lack funds to acquire hardware and software dedicated to teaching. The pedagogical consequences are an increasing sense that our students (and by this we mean those in liberal arts programs) are falling behind those in the undergraduate professional schools. This is happening just as we are attempting to win back some ground we lost in the seventies by showing that a liberal arts education is better than an undergraduate education in business or in "communications."

The project we describe in this paper is an experiment designed to show how liberal arts students can receive an intensive introduction to microcomputers while simultaneously doing intellectually stimulating work. The structure of the project is a year-long junior seminar given in the Undergraduate Honors program in the College of Liberal Arts of the University of Texas at Austin.(3)We are doing what our title suggests: reconstructing the lives of the inhabitants of a historical community, in this case Fredericksburg, Texas, from the mid-nineteenth century until 1910.

Fredericksburg is a town well-known to most Texans. It is located in a semi-arid hilly region about 75 miles west of Austin.(4) It is a German town, founded by a German settlement society in 1846, and strongly German even today. Its coherent ethnic culture (even if pierced fairly quickly by Anglos and somewhat later by Hispanics) make it an ideal choice for the direct study of demographic and social characteristics, and the indirect study of genetic characteristics. Moreover, the town's strong sense of its own history and culture made it possible for us to acquire many documents which other communities have not kept and often will not share with outsiders. Church records of baptisms, marriages and burials are especially important to us. Fredericksburg had its own organized churches from the first days of settlement, and virtually all their oldest records survive. Moreover, we received wonderful cooperation from all the churches, so that we have been able to use at least three–fourths of all the church records ever kept.

Our goal has been to use the reconstruction of the lives of a town's inhabitants as a means to get students involved in a team research project using original source documents and making sophisticated use of microcomputers. Our nine students are building the data base and working with Gutmann on programming. Each is writing a paper due this spring in which he or she examines a single topic about Fredericksburg. These papers cover the basics of demographic history (fertility, nuptiality, mortality and migration); the social, economic and ethnic–racial structure and evolution of the community; patterns of inbreeding in this ostensibly "closed" community of Germans; and the relationship between the houses of Fredericksburg and the lives of the families that lived in them.

We expect the data base we construct to have a life of its own. Already several of our students have said that they will add to the basic data we have collected this year and write more complex senior honors theses next year, on medicine and health, on marriage, family and inheritance, and on social organization. Later, these data should be available for further analysis by other students. Finally, we will start the cycle again next fall by offering the course again, in order to build another data base with a year's experience behind us.

Our general computing strategy was to do with microcomputers what had been done in the past with pencil and paper or with mainframe computers, and to do it with standard database, spreadsheet and statistical analysis software. This is possible because the most recent generation of microcomputers with their fast and capacious disk drives allow us to store a large quantity of data. We expected to use the mainframe only to upload backup files, and perhaps to print large jobs (in the end, neither use of mainframes was practical). We saw the historian's data base which consists of a variety of information about an individual or a family as a special case of the relational database used by businesses, so that it no longer called for special programming in a high level language like basic, fortran or cobol. Thus we were less concerned about writing a general purpose tool than for fitting our problem to a general–purpose tool already in existence. By choosing commercial software for the microcomputer and by eliminating mainframe use with its barriers in the form of complex terminal commands and mainframe operating systems, we expected to make our students able to conceptualize a problem, turn on the computer, pick the program required from a menu of choices, and go to work. We have achieved this goal.

2. IMPLEMENTATION

Our data consist of the following, and their handling dictated the implementation of our computing strategy:

1. Censuses taken in Fredericksburg every ten years from 1850 until 1910, with the exception of 1890 (which were destroyed in a fire before they were microfilmed). The census lists every household, and within every household every person, as well as their characteristics. Some of these characteristics are simple, such as age, color, sex, marital status, and literacy, while others are complex and invaluable, like the question on the 1900 and 1910 censuses which asked every woman how many children she had ever born, and how many of them were still alive. In our original pass through the data we identified about 10,000 census entries for Gillespie County of people who were residents of Fredericksburg.

2. Lists of children baptized, couples who married, and people who were buried by each of the seven churches in Fredericksburg (4 Lutheran, 1 Catholic, 2 Methodist). Such records often name other persons involved, such as parents, as well as godparents and witnesses. They also can provide information about each person mentioned, such as place of birth. We made use of about 9500 baptisms, about 2000 burials, and about 1600 marriages. These 13,000 events name some 55,000 people, of whom 20,000 were godparents at baptisms. To these religious records we have added data drawn from the civil marriage licenses (available from 1850–1910), and from civil birth and death certificates (available from 1903–1910). We have also begun to build a file of exact birthplaces for the immigrant population, in order to sustain our study of migration into Fredericksburg.

In the future we hope to add information drawn from annual ad valorem property tax roles, from county court probate records, and from the files describing property transactions in the Gillespie County Courthouse in Fredericksburg. These records will enhance our knowledge of the people and Fredericksburg, and their personal behavior.

Our computing environment consists of two PC/ATs and one PC, linked into a PC Network. The two ATs each have double 20–megabyte hard drives; the PC has an external 10-megabyte hard drive. All three computers have Enhanced Graphics Adapters and enhanced Color Monitors. Printers have been the weak link in our operation thus far. We have rather large printing requirements for the construction and maintenance of the data base. We have made use of two IBM Color printers, but they have not proven satisfactory. We have now requested a Page Printer, which should solve our problems. Finally, two computers are hard-wired to the University of Texas' network of mainframe and mini computers. Our most important software has been dBASE III for data management, Microsoft Word for Word Processing, and SYSTAT for statistics.(5)

We have entered data by giving students photocopies or microfilm (we have two microfilm readers alongside the PCs), and allowing them to key data directly to dBASE III, using a simple codebook. Most of our early data entry was done using the DBASE III Append command, which allows the direct addition of records to an existing file structure; more recently, as we have worked with a file linking many kinds of records, we have used custom input screens. The data entry process went quickly, and we have been exceptionally satisfied with the quality of data entry we achieved using this method.

The next step after data entry was to link together all the references to a single individual or to a single family. There is considerable debate among scholars about how best to do this, with divisions coming between those who advocate wholly automated record linkage systems (in which all linkage decisions are made by machine), and those who advocate semi-automated systems (in which the computer does a variety of sorting and consolidation steps, but the researcher makes the linkage decision).(6) We decided on a semi-automated system, in part because the data we used contain a very large quantity of name variation. This name variation is the result of the variety of records we have, with Censuses taken by outsiders in English, and Church records kept by local priests in German and in English, for a population that spoke German, English and Spanish (so that Wilhelm Friedrich Ahrends can become Billy Bob Ernst). Automatic linkage is very difficult without linguistic uniformity.

We generated alphabetized lists of all the persons named in all our records (except for baptismal godparents), some 45,000. We divided these lists into eleven sections, and spent a month deciphering names and assembling linkage lists. The result was a list of about 18,000 individuals who lived in Fredericksburg between 1846 and 1910. The data are still organized in the original coding format, but they are now linked by three indexed linkage files which tell us which data items refer to each individual, and which refer to each family. These three linkage files constitute the heart of the data retrieval system we use to exploit the data base we have constructed.

In its current condition, the Fredericksburg data base thus consists of one file for each of six censuses, and one file each containing information about baptisms, marriages and burials. In addition, we have three "linkage" files for locating information about individuals and families. These are all dBASE III files, and they occupy about 14 megabytes of hard disk storage, which we keep on one PC/AT, with a backup copy on the second. We have plenty of room for expansion.

We use the data we have accumulated in two ways, in the form of census lists based directly on the coded censuses, and in the form of "life histories" of individuals or families, assembled from all the elements in the data base. The census lists can be easily analyzed for characteristics of the population at the six time points by use of the SYSTAT programs. We have created SYSTAT system files, and all of our students have become quickly adept at drawing out contingency tables and other statistics.

The information retrieval software which we use to create life histories is all written as dBASE III programs, and all the variants do essentially the same thing. These programs take a single individual or family, and produce a life history. For data checking and demonstration purposes, the programs put the life history on the monitor's screen, or send it to the printer. For purposes of quantitative analysis, the programs extract the information necessary for analysis (for example, a woman's fertility history, or all the ages at first marriage in the population, or all the ages of death), and put them in a simple file for analysis, either with dBASE III programs or with a spreadsheet (we have Planning Assistant and expect to order Lotus 123 soon), or SYSTAT. These programs process a single person or family or a small number of persons or families quite quickly. Working through all of this large database can, on the other hand, take a considerable amount of time.

3. RESULTS

This project has been a success. We have a working database system with interesting data, and we have captured the attention and the energies of our students. In an academic environment where levels of student diffidence can be very high we have an exceptional level of commitment. We set out to create a team research environment in order to get a lot of work completed. We have managed to create that environment. The risk, in fact, is that we will take too much from our students, because the project is so attractive and their relationship with faculty so close. They have a very hard time saying 'no' to us when we ask them to do additional work.

The research results we are getting are of a consistently high quality, which means we should be able to produce a series of professionally publishable papers. This is a clear achievement of our goal to show students how to do mature work in a realistic environment, without taking them out of a liberal arts program where they can pursue a broad range of interests without penalty. To put the matter simply: we have shown them how to be professionals without making them professionals in this field.

We have not achieved these results without some costs and mistakes, and we hasten to point these out to any who would emulate us. First, the project has been expensive, in terms of time and materials. It has taken two of us, working far harder than one normally does to teach undergraduates, to organize and teach this course. We have required a very substantial computer system, one which is dedicated to this one course and not shared in a computer 'lab'. We have spent a great deal of money on software, and expect to spend more. In addition, our requirements for diskettes, paper and printer ribbons, and the cost of obtaining copies of the documents we have used thus far, put a considerable strain on the funds available for new course development. Our administration has been exceptionally far-seeing in supporting these expenses. They may not be great in the context of teaching the sciences, but in a history curriculum where lecturing predominates and the cost of instruction is limited to gradebooks, red pens and ditto paper for exams, we have spent a lot. Should many courses like ours be offered, our visionary Dean, Director and Chairman may have to add magical skills to their repertoire.

Second, we made some mistakes in organizing the research. Our original unit of analysis for this project was the town of Fredericksburg. We learned later that by incorporating the church records, our field of study should have been the whole county or the church membership, not the town. This has forced us to expand the scope of our work, and to conclude that Fredericksburg was too large to study in a single year. Next year we will work on a town that is smaller, or a whole county that is smaller than the county in which Fredericksburg is located. We have worked feverishly for six months to incorporate basic data into our data base. With a smaller field of study, we can include more varieties of data, and ask more––and more interesting––questions.

We set out to show that existing commercial technology and software can be used unaltered to do scholarly research and to teach the techniques of computing and research to students. That this is possible is clearly evident, although at a cost greater than that of normal teaching. The experience has been exceptionally rewarding.

REFERENCES

Bouchard, G., R. Roy and B. Casgrain. 1985. Reconstitution utomatique des familles. le système SOREP. Chicoutimi, Quebec: Centre Interuniversitaire de Recherches sur les Populations.

Dyke, B. and W. T. Morrill. 1980. Genealogical Demography. New York: Academic Press.

Fleury, M. and L. Henry. 1965. Nouveau manuel de dépouillement et 'exploitation de l'etat civil ancien. Paris: I.N.E.D.

Fliess, K.H. 1986. The Historical and Genetic Demography of the Wends of Serbin, Texas. Unpublished Ph.D. Dissertation, Department of Anthropology, The University of Texas at Austin.

Goubert, Pierre. 1960. Beauvais et le Beauvaisis. Paris: S.E.V.P.E.N.

Greven, Philip. 1970. Four Generations. Ithaca: Cornell University Press.

Gutmann, M.P. 1977a. Reconstituting Wandre: An Approach to Semi-Automatic Family Reconstitution. Annales de Démographie Historique, pp. 315–341.

Gutmann, M.P. 1977b. The Future of Record Linkage in History. The Journal of Family History 2: 151–158.

Gutmann, M.P. 1980. War and Rural Life in the Early Modern Low Countries. Princeton: Princeton University Press.

Gutmann, M.P. and Randy Wyrick. 1981. Adapting Methods to Needs: Studying Fertility and Nuptiality in Seventeenth- and Eighteenth-Century Belgium. Historical Methods 14:163-172.

Henry, Louis. 1956. Anciennes familles genevoises. Etude demographique: XVIe-XXe si cles. Paris: I.N.E.D.

Henry, Louis, 1980. Techniques d'analyse en demographie historique. Paris: I.N.E.D.

Jordan, Terry G. 1966. German Seed in Texas Soil. Immigrant armers in Nineteenth-Century Texas. Austin: University of Texas Press.

Macfarlane, Alan. 1977. Reconstructing Historical Communities. Cambridge: Cambridge University Press.

Meinig, D.W. Imperial Texas. Austin: University of Texas Press.

Swedlund, A.C. 1971. The Genetic Structure of an Historical Population: A Study of Marriage and Fertility in Old Deerfield, Massachusetts. Research Report No. 7, Dept. of Anthropology, University of Massachusetts, Amherst.

Winchester, Ian. 1973. On Referring to Ordinary Historical Persons. In E.A. Wrigley, ed., Identifying People in the Past. London: Edward Arnold.

Wrigley, E. A., Family Reconstitution. In E. A. Wrigley, ed., An Introduction to English Historical Demography. New York: Basic Books.

FOOTNOTES

1 For a good theoretical discussion, see Winchester, 1973.

2 The intellectual origins of this kind of work among historians and demographers are found in the French demographer Louis Henry's invention of the technique called "Family reconstitution," (Henry, 1956, 1980; Fleury and Henry, 1965), as developed by historians interested in a wider range of subject matter. Among many works in this area, the following are pioneering: for England, Wrigley (1966), and MacFarlane (1977); for France, see Goubert (1960); for the U.S., see Greven (1970). For our work in this area, see Gutmann (1977a; 1980; 1981), and Fliess (1986). Among anthropologists and geneticists micro-level studies have a long history. See, again among many others, Swedlund (1971), and Dyke and Morrill (1980). For a combination

of disciplines, the work of the Consortium based at the University of Quebec at Chicoutimi under the direction of Gerard Bouchard, is probably the most successful. See the references in Bouchard, 1985.

3 This project could not possibly have been conducted without a large quantity of institutional cooperation and assistance. In addition to thanks we owe to Project QUEST which provided hardware and software, we are indebted to Professor Standish Meacham, chair of the Department of History; Ellen McDonald and Professor Ira Iscoe of the Plan II Honors Program; and Dean Robert King of the College of Liberal Arts for moral and financial support.

On the cultural and geographical development of Texas, see Meinig (1969). On Germans in Texas, the work of Jordan is most important. For an example, see Jordan (1966).

5 We had originally hoped to use Statistical Analysis System (from SAS Institute, Inc.), but it was not available in time. We are now evaluating statistical packages more thoroughly.

6 For one point of view, see Gutmann (1977b). For a report on a very successful automatic strategy, see Bouchard, et al (1985).

SAMP – A PROGRAM TO COMPUTE SAMPLE SIZE

Presenter: Peter A. Lachenbruch *University of California, Los Angeles*

In elementary Biostatistics classes, the importance of planning experimental and observational studies is stressed. Various aspects of this effort include design of questionnaires, pretesting of all materials, appropriate allocation of subjects to treatments, interviewers, etc. and choice of a number of observations which will enable the researcher to answer the questions being posed (This of course is not an exhaustive list). Some of this work can be accomplished with word processing equipment and field work and draws on verbal skills. Proper formulation of the scientific questions draws on the subject matter expertise of the investigator.

Determining the sample size for a study depends on a number of factors which are related to the statistical procedure being used. These involve a moderately complicated amount of arithmetic. For example, in order to compute a sample size for a two sample binomial test, the student must decide

a. Significance level
b. Power (chance that a given difference will be detected)
c. Probability of an event in the first group
d. Probability of an event in the second group
e. The ratio of the samples in the two groups

These numbers are then combined using familiar formulas found in all elementary texts. I have found it very useful for the students to compute a number of sample sizes, varying one or more of these factors each time. In this way, the students can learn how the various factors affect the sample size requirements. However, most students (and others!) find that such calculations are boring, prone to errors, and fail to reach an adequate understanding of the sample size computations. In addition, in practical situations, the choice of various combinations of these factors may lead to unreasonable sample size requirements. It is important that these be recognized. For example, planning a study of Hodgkin Disease which requires 500 cases is unlikely to be possible in any single hospital. The ability to recompute sample sizes easily thus becomes a useful practical tool.

My hope was to provide an easily used program which would compute sample sizes for a variety of cases which are studied in elementary courses. Such a program would allow the student to choose a procedure, enter the parameters, and observe the sample size requirements. They should then be able to retry the procedure while modifying only one parameter. Thus, the previously specified inputs should appear as defaults. The students would be able to explore the consequences of various parameter combinations, and be able to concentrate on the implications of their parameter choices rather than on the arithmetic. In addition, some fairly advanced statistical methods are included, so that this program has relevance to later courses.

The School of Public Health is developing a Microcomputer Instructional Laboratory to integrate the use of microcomputers in our teaching. The laboratory consists of 24 IBM XTs (without hard disks) networked with 4 IBM PC/ATs. We have networked versions of a number of programs and languages, as well as a number of stand alone programs. The SAMP program was developed on an AT using the DeSmet C compiler. Thus, it should be easily portable to other computers.

The program was completed after about 3 man-months of development. The majority of the effort was in debugging the input routines. In the C language, if one uses the scan function to enter a numeric quantity, but a non-number is entered, the program behaves in a very unfriendly manner. Therefore, I decided to read all entries as strings and convert them to numbers if they were acceptable. In so doing, I managed to create new problems (it was a learning experience for me). The program allows the user to perform the same computations repeatedly using previously entered quantities as defaults. The output consists of three sample sizes: one for the parameters as requested, one for a closer alternative hypothesis (i.e. more difficult to detect), and one for an alternative that is farther away. These give a range of values which guides the user in new parameter combinations. The usual sample size computations that students are given include one and two sample binomial tests, one and two sample normal tests, and binomial and normal confidence intervals with fixed lengths. These capabilities are included, and in addition, fixed number of standard deviations confidence intervals, matched pair designs for binomial and normal cases, nonparametric tests (sign, signed-rank, Wilcoxon), and log-rank test are given. Most two sample tests are offered with equal sample sizes. However, in some situations it is easy to obtain many observations in one group, and difficult to obtain them in the other. In such a case, there is good reason to use unequal sample sizes. This is possible by specifying that the ratio of sample sizes is other than 1.

A brief description of the program is given together with the input requirements. (Notation: A = Significance Level, W = PoWer, C = Confidence Level, P0 = null probability, P1 = alternative probability, D0 = Difference to be detected, S = standard deviation, R = ratio of sample sizes, H = hazard ratio, F = Fraction failing, OR = Odds ratio, M = matchable fraction, D = probability of a discordant pair, L = length of Confidence interval, K = number of s.d. for fixed length confidence interval)

Distribution	Input Parameters	Alternative Method
One Sample Binomial	A, W, P0, P1	Multiply Log Odds
Two sample Binomial	A, W, P0, P1, R	Multiply Log Odds
Binomial Confidence		
Interval: Fixed	C, P0, L	Multiply L
Relative	C, K	Multiply K
Matched Pairs		
Binomial	A, W, OR, M, D	Multiply OR
One Sample Normal	A, W, D0, S	Multiply D0
Two Sample Normal	A, W, D0, S, R	Multiply D0
Normal Confidence		
Interval: Fixed	C, S, L	Multiply L
Relative	C, K	Multiply K
Matched Pairs		
Normal	A, W, D0, M	Multiply D0
Log-rank	A, W, H, R, F	Multiply log(H)
Non-parametric		
Sign test	A, W, P1	Multiply Log Odds
Signed Rank	A, W, P1	Multiply Log Odds
Wilcoxon	A, W, P1, R	Multiply Log Odds

The final product had a few changes from the program as originally planned. First, it was necessary to make the program stable to "unreasonable" input values. The program originally protected against illegal values (e.g.

probabilities less than 0 or greater than 1), but it was found that some students would enter values which gave extremely large sample sizes (greater than 1,000,000) and these numbers couldn't be relied upon. Therefore, the legal parameter values were restricted so that sample sizes greater than 100,000 generate an error message. At the start of the programming effort, the sample sizes for the non-parametric procedures were not available. A paper appeared giving these in a simple form, and they were included.

These procedures cover only one and two sample tests. There is very little new in the way of statistical theory here. One of the major problems that statisticians face is computing sample size in Analysis of Variance problems. The mathematics is available, but it is messy. Another program in this set will compute power for various configurations of means and standard deviation. Other statistical procedures which are programmed include: Cumulative Normal distribution, Normal percentiles, Cumulative Gamma distribution, Gamma percentiles, Cumulative Beta distribution, Beta percentiles, Discrete Distributions (includes Binomial, Negative Binomial, Poison, and Hypergeometric), Cumulative Weibull distribution and percentiles, and Randomization.

TOOLS FOR LEARNING DATA ANALYSIS

Presenter: Eugene Kroch *University of Pennsylvania*

copyright © University of Pennsylvania, 1986

INTRODUCTION

Students are often confused by the abstractions of statistics textbooks. The complicated mathematical formulas used to characterize data– generating processes intimidate the typical student primarily because they combine probability theory with limit processes and algebraic manipulations which provide little intuition to the student. Even when a text provides concrete examples, students still may not develop the ability to apply the methodology to new circumstances. The microcomputer is a means to experiment with statistical techniques and data sets in order to help students develop an operational knowledge of methodology. The computer can free students from the tedium of hand calculations and provide them with a guide to the application of techniques. The microcomputer can be much more than a sophisticated calculator. It can be a teacher that checks the student's comprehension of new material, corrects interactive exercises, and requires the student to respond correctly to questions before advancing to the next stage of the course. This view of the microcomputer motivates our efforts at the Undergraduate Data Analysis Laboratory at the University of Pennsylvania.

The wide variety of courses at the College that teach or employ quantitative methods can be divided into three principle categories: (1) advanced topics, (2) introductory principles, and (3) application modeling. Each of these course types has its distinct teaching environment with corresponding demands on the computer. The first category applies to quantitative topics courses which apply statistical techniques to a range of related problems. These courses may focus on fields outside of statistics in the social or natural sciences but they assume a background in mathematical statistics. Here the role of the computer is to provide a set of "packages" that will permit the students to perform calculations with high computational requirements. These packages can be commercially available software or home–grown programs tailored to the specific needs of the course. At Penn we have some programs (APL workspaces) that provide an interactive environments that match statistical techniques with application data sets.

The second type of teaching environment is exemplified by the introductory or "principles" course (although it is by no means restricted to it). Here the ideal role of the microcomputer is to provide an "electronic textbook." It is quite literally a textbook, but much more than just the transfer of words, formulas, and diagrams to the screen from the printed page. The electronic textbook, going beyond the constraints of the printed textbook, can pose questions to check the student's comprehension. If the student fails to answer correctly, the program will coach him until he finds the answer. Based on the students responses the program may route the student on alternative paths through the subject matter. In other words progress through the course is be self–paced or more accurately "computer– paced." Finally, at the end of each chapter the student is tested. The objective portion is computer graded and the essay portion is left for the instructor to grade. The advantage of having students answer even the essay questions in this manner is that (a) it is a convenient way to store and to update the questions and (b) it provides a word processor to the student. A randomizer is used to select questions from a large test question bank. Since each student will answer a different set of objective questions there is little incentive to collaborate or to repeat chapters in order to improve scores. The UDAL at Penn has developed a total of ten chapters for electronic textbooks in introductory microeconomics and macroeconomics. We are currently working on electronic texts for introductory statistics and econometrics.

THE PROGRAM

It is the third teaching environment that motivates the specific set of tools presented in this forum. Applications modeling occupies a niche somewhere between the advanced topics of the first category and the introductory principles of the second. Indeed, it is meant to bridge the gap between theory and practice and may be found sometimes (although rarely) in courses that are primarily in one of the first two categories. The programs are designed to give students a feel for data both descriptively and inferentially. In the descriptive portion they can experiment with methods of summarizing and displaying data. One of the most important lessons is in the effects of functional transformations on the shape a data distributions and how re–expressing data can dramatically alter the impression the make on the researcher. In the inferential portion students use Monte Carlo methods to confirm and demonstrate that a class of estimators in practice has the properties that have been derived analytically in the textbook. They can experiment with the effects of sample size especially where small sample properties can not be expressed mathematically. They can see how ordinary least squares estimation can be deficient when certain conditions of the Gauss–Markov theorem are violated.

Using APL and a network of IBM PCs, we have been able to create an extremely friendly interactive environment where students are guided through the application of statistical techniques introduced in classroom lectures. The system is fully menu driven and organized to be compatible with the textbook and the pace of the lectures. For example in Economics 620, Data Analysis and Econometrics, students are presented with the following menu:

THE COLLEGE

UNIVERSITY OF PENNSYLVANIA

UNDERGRADUATE DATA ANALYSIS LABORATORY

ECONOMICS 620 TEACHING DISK

Choose one of the following chapters to run:

EDA CHAPTERS

1. Stem–leaf and Boxplot
2. Comparisons
3. Relating Y to X
5. Two–way Tables
6. Smoothing
7. Robust Fitting

REGRESSION AND TESTING

8. Nonparametric Tests
9. Fitting Distributions
10. Monte Carlo Simulations
11. Regress I – Demand Analysis
12. Regress II – Macro Models

Enter the number of the chapter you wish to run:

Using APL and a network of IBM PCs, we have been able to create an extremely friendly interactive environment where students are guided through the application of statistical techniques introduced in classroom lectures. The system is fully menu driven and organized to be compatible with the textbook and the pace of the lectures.

EXPLORATORY DATA ANALYSIS

In the section on data description students work examples using exploratory data analysis: stem–and–leaf plots, box plots, smoothing, robust fitting, and other methods, all using APL's graphics features. They are then asked to choose two or three of their own applications from data sets and techniques provided. A workbook accompanies all exercises, asking questions and providing feedback. A couple of pages from Chapter 1 of the workbook (Data Displays) are included in the appendix. They show how a box plot is constructed and how alternative transformations of data, displayed in stem-and-leaf form, can help answer questions about the data set. The following is an example of the screen that appears when the student selects option 2 from the menu:

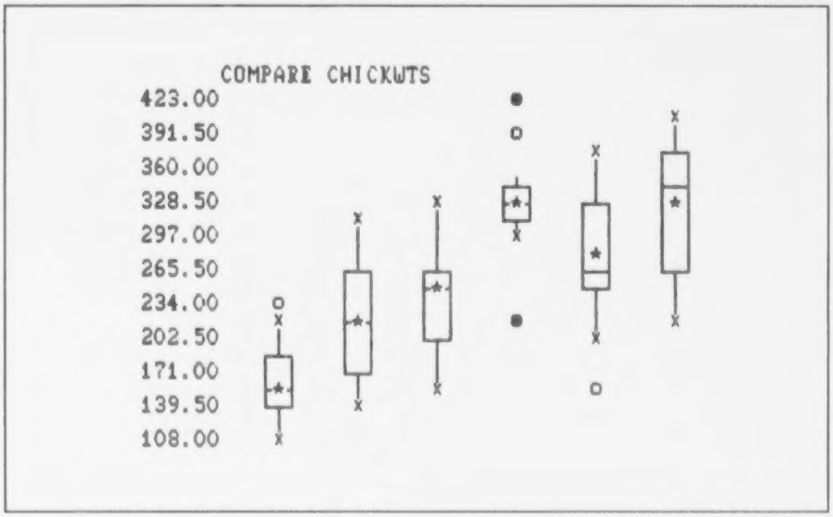
Chapter 2 Loaded -- COMPARISONS
See DESCRIBE PF10 returns menu
Chapter 2: COMPARISONS

These are the vectors used in Chapter 2 for the analysis of multiple batch data sets. The first two are used in the demos....Please note that some of the vectors are of unequal lengths, and therefore contain some zeros.

WARPBREAKS	The number of warpbreaks on 9 looms for 6 yarn types.
CHICKWTS	The weight in grams of 6-week old chicks in 6 groups, With missing values changed to zero.
DIABETES	A matrix made for Problem 1, consisting of the columns NORMAL, ALLOXANDIABETIC, and TREATINSULIN
SYLLABLES	The matrix for Problem 2 in 5 groups (columns).
PRUSSIANS	The matrix for Problem 3, containing Prussian infant death ages in days for MALELEGIT, MALEILL, FEMLEGIT, FEMILL
BIRTHRATES	The matrix for Problem 4, birthrates for selected years of women in 6 age groups (columns).
ILLITERACY	The matrix for Problem 5, with regional illiteracy rates over 6 years (columns).
AIRCRASH2	The matrix for Problem 6, by month and year. HINT: use FORM

Procedures are COMPARE, CONDENSE, AND SCAT. Outputting control parameters are WIDTH, DEPTH, NGAP, NUM, and MISS.

If the student then applies the COMPARE procedure to the second data set listed, he will get the following set of parallel box plots:



ESTIMATING AND TESTING

In the section on inferential techniques, students explore the properties of estimators and testing procedures. They begin with nonparametric testing in Chapter 8 and then in Chapter 9 learn how to judge which ideal distribution best characterizes a given sample. In Chapter 10 students run a series of Monte Carlo experiments designed to demonstrate how these inferential methods differ from data description techniques. Results of simulations are displayed graphically. Again, students can control the environment and choose applications from a menu of options. In the final chapters, students specify, estimate, and simulate a multi–equation models. An accompanying set of exercises help the student to explore its dynamic properties.

If the student selects Chapter 10 on Monte Carlo simulation, he will be greeted by the following screen:

Ordinary least squares regression gives parameter estimates that are BLUE (best linear unbiased estimates) provided that certain conditions are met. These conditions are:

(a) the relationship between the dependent and the explanatory variable is linear

(b) the explanatory variables are nonstochastic and, hence, are uncorrelated with the error term

(c) the error term has a mean of zero

(d) the error term has constant and finite variance

(e) the error term is not serially correlated

If any of these assumptions does not hold, the parameter estimated may not be BLUE.

...PRESS ANY KEY TO CONTINUE...

Let us explore what might happen if either condition (d) or (e) does not hold. The first would give us heteroscedasicity and the second would generate serial correlation. To show the resulting effects on our OLS estimating procedures, we shall generate sampling distributions for our parameter estimates and display their properties. In particular, we can compare the means of these sampling distributions with the "true" (predetermined) parameters to see if there is bias. We can look at the standard deviation of the sampling distribution to see how efficient the estimates are and compare this "true" efficiency with the estimated standard error that we calculate for each coefficient.

To examine the effects of not meeting OLS conditions for BLUE parameters type REGSIM to enter a simulation procedure. This simulation assumes that you know the "true" parameters (the real intercept, slope, the mean and standard deviation of the independent variable, the standard deviation of the error terms, the number of observations per iteration, and the number of iterations that were run). In addition, for the heteroscadastic and the autocorrelated models it is assumed that you know the associated variability weight and degree of autocorrelation, respectively.

If the student then issues the command REGSIM, he could have the following interactive session:

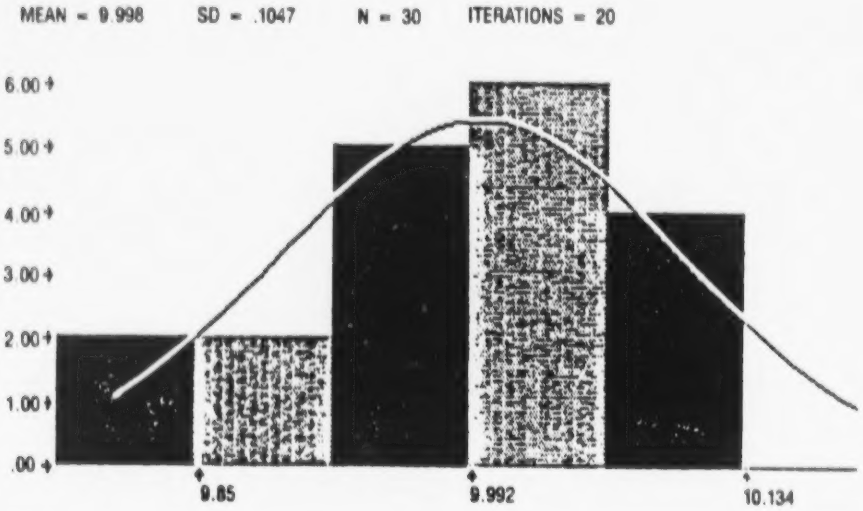
```
REGSIM
What is the linear relationship between the independent and dependent
variables ?  (Specify an intercept and a slope, in that order.)
-1500 10
How many iterations do you want?
20
What type of regression simulation do you wish to run ?(Normal, Uniform,
Autocorrelated, Heteroscedastic, or STOP)
A
What is the standard deviation of the error term?
 50
What is the value of RHO that you wish to use in this simulation?
.5
What sample size do you wish for this simulation?
30
----------------------------------------------------------------
ITER ALPHA   STD ERROR  BETA    STD ERROR  STD ERROR   R2
  #   HAT     ALPHA HAT  HAT     BETA HAT   REGRESS
----------------------------------------------------------------

1         -1495.25  15.70     10.08887  0.28108   48.378    0.979
2         -1482.46  16.47     9.69674   0.29481   50.742    0.975
3         -1516.51  18.29     10.18049  0.32746   56.360    0.972
4         -1488.41  15.83     10.16953  0.28330   48.761    0.979
5         -1439.44  20.06     9.17216   0.35916   61.817    0.959
6         -1464.74  12.86     9.39001   0.23024   39.628    0.983
7         -1455.71  17.73     9.12721   0.31746   54.640    0.967
8         -1516.28  15.82     9.99187   0.28323   48.748    0.978
9         -1456.40  20.69     9.31089   0.37044   63.759    0.958
10        -1506.47  17.31     9.96101   0.30988   53.335    0.974
11        -1501.04  21.99     9.68991   0.39372   67.766    0.956
12        -1427.53  15.37     8.81041   0.27506   47.342    0.973
13        -1494.37  15.16     9.54487   0.27144   46.720    0.978
14        -1441.14  13.28     9.46648   0.23772   40.915    0.983
15        -1543.43  16.09     11.04207  0.28811   49.589    0.981
16        -1443.30  18/35     9.36403   0.32854   56.546    0.967
```

```
17         -1546.60  13.28      10.21871  0.23775   40.920     0.985
18         -1461.26  17.67      8.96354   0.31634   54.447     0.966
19         -1459.47  14.95      8.97589   0.26766   46.068     0.976
20         -1490.62  14.26      9.53174   0.25521   43.925     0.980



                                          ALPHA            BETA

MEAN OF ESTIMATES OF COEFS :  -1481.52                     9.63482
STANDARD DEVIATIONS OF COEFS  :  33.35                     0.53260
MEAN STANDARD ERRORS OF COEFS :16.56                       0.29643
MEAN STANDARD ERROR OF REGRESS :   51.020
MEAN R-SQUARE :                                  .973
```

Which of the following do you wish to see graphed?

1) The intercept estimates
2) The standard errors of the intercept estimates
3) The estimates of the coefficient for the independent variable
4) The standard errors of the coefficient's estimates
5) The standard errors of regression
6) None of the above

Choosing option 3 would yield the following graph:



The student can evaluate the symmetry of this sampling distribution and its resemblance to the gaussian ideal. He is, of course, expected to work through a number of these simulations, exploring the effects of sample size, the degree of autocorrelation or heteroscedasticity, and the magnitude of the error variance. Returning to the main menu, the student may choose other chapters that set up estimating and testing environments for demand analysis (Chapter 11) or for macroeconomic modeling (Chapter 12).

APPENDIX

EXPLORATORY DATA ANALYSIS

In the array PRECIPITATION (69 observations) the median is such that 69/2 - 34.5 of the numbers are on either side, so the median is the 35th number in the ordered batch. From the stem-and-leaf plot this number would be 36. The lower quartile is such that 69/4 - 17.25 numbers are below it. This is the 18th number in the ordered batch, and therefore 29. The upper quartile is the 18th largest number, which is 43. These are plotted in figure (E).

Next, the program draws a rectangular box, with ends corresponding to the lower and upper quartiles. The program also replaces the median with an asterisk. The length of the box is the interquartile distance. Crosses mark the highest and lowest data values that fall between these observations, and any values whose positions are outside these crosses are marked with circles, those more than 1.5 times the interquartile distances outside each quartile get heavy circles.

```
(E)  -------------------|-----|-----|-----------------

                           29  36    43

(F)                      !----------!
        -------|----------!   *   !---------|------
                         !----------!
            15                                57

                        !----------!
(G)           x----------!   *    !---------x
                        !----------!

                          !----------!
(H)        oo ooox---------!  *     ! ---------xoo  o
        22                !----------!
```

The ten outliers in figure (H) are the two 7's, the two 8's, 12, 13, 14, 59, 60, 67, the first two and the last one getting heavy circles.

If you enter the term:

BOXPLOT PRECIPITATION

the computer responds with the following display:

```
  7.00         !----------!           67.00
oo ooox---------!      *     !---------x o o   o
22              !----------!           2
```

NB: The LENGTH parameter governs the size of BOXPLOT output. See what happens when you vary LENGTH between 30 and 60.

TRANSFORMATION OF DATA

Consider the following batch of numbers. They represent the areas in square miles of the 50 largest bodies of land. Some areas are considered islands, and others, continents. We want to consider whether Australia is more like an island or a continent. (Areas measured in 1000's):

Axeld	16	Baffin	184	Banks	23	Devon	21	Elles	82
Melv	16	P of W	13	South	16	Vict	82	Nov Z	32
Spits	15	Gt Br	84	Ireld	33	Green	840	Ice	40
Newf	43	T d. F	19	Cuba	43	Hisp	30	Ceylon	25
Madag	227	Taiwan	14	Hainan	13	Hokk	30	Honshu	89
Kyushu	14	N.Guin	306	NZ(N)	44	NZ(S)	58	Luzon	42
Mind	36	Sakh	29	Tas	26	Vanc	12	Brneo	280
Celeb	73	Java	49	Moluc	29	N. Br	15	Sumat	183
Timor	13	Asia	16988	Afr	11506	N.Am	9390	S.Am	6795
Eur	3745	**Aust**	**2968**	Antar	5500				

(We have boldfaced the entry for Australia)

If we request a stem-and-leaf plot entry for Australia:

```
STEMLEAF ISLANDS
```

the computer responds with the following display:

```
00|000000000000000000000000000000111111222338
02|07
04|5
06|8
08|4
10|5
12|
14|
16|0
```

Because of the skew in the data produced by the large continents (especially Asia), this display is not very informative. We could take the square root by typing:

```
SRILS <-- ISLANDS * 0.5
```

and then, `STEMLEAF SRILS`

Does this help?

Even better would be to take the common logarithm by typing:

LNILS <-- 100 ISLANDS
and then, STEMLEAF LNILS

for which we get,

```
01|111111222223344
01|5555556666667899999
02|3344
02|59
03|
03|5678
04|012
```

Now what do you conclude?

THE GREAT AMERICAN HISTORY MACHINE

Presenter: David W. Miller *Carnegie–Mellon University*

I would like to begin with an observation: Computer people often express surprise when they learn that I am an historian, but historians are almost never surprised when they learn that I work with computers. During the past fifteen years or so, elementary statistical methods have become a common, if not an absolutely universal part of the historian's tool kit. To be sure, many historians of my generation and its predecessor still find computers, and indeed anything to do with numbers, to be deeply mysterious and fundamentally alien. However, for younger historians, some experience with computers is routine.

Nevertheless, apart from word processing, which has delivered our profession from the toils of staples, scissors and rubber cement, I would argue that computer applications remain near the fringes of most historians' work, even that of many who have acquired computer literacy. This is not because historians lack quantitative aptitudes, though some do. Indeed, I would guess that in quantitative sophistication historians are fairly representative of professional people generally. Very few are as sophisticated as physicists and recently–minted engineers, but most are at least as numerate as typical managers and physicians. Put slightly differently, they are very similar on this dimension to other members of the market for personal computers which IBM and other manufacturers discovered a few years ago.

The reasons why computers have not become as central to the historical profession as they have to some other disciplines, in my view, is that until recently the hardware and software accessible to most academics was simply not much help with the central tasks of the historian (again, except for the task of writing). Let me explain.

Those historians who began to do quantitative work about twenty years ago generally got their ideas as to what computers could do from other social scientists –– economists, sociologists and political scientists. These disciplines had their own research agendas driven by theory. What their practitioners wanted from the computer was help in the testing of hypotheses derived from theories. Certainly the testing of hypotheses is a part of the historian's agenda, but usually he or she does not start from a formal theory. Historians typically proceed by looking a large amount of evidence –– a very large amount, by the standards of other disciplines -- and then trying to discern patterns in the evidence. There has been very little in the accessible computer world which seemed to help with this task, so many historians have thought of computation as a tool which, like the foreign language which, say, a U.S. historian was required to work up in graduate school, might be placed on a shelf to be dusted off if needed, but not firmly attached to the workbench for daily use.

The utility which I will be describing today is one effort to use the computer to make a large amount of evidence accessible in a way that allows one to discern patterns -- to "make sense" of it. I hope that I can persuade you that while this software is of particular interest to historians, the type of thinking it facilitates is common to a wide spectrum of thoughtful people and that its implications are not confined to the historical profession.

Now let me turn to the specific educational problem which led to this project. For some years now in the CMU history department we have had one or more computer exercises as normal components of our introductory history courses. These exercises have generally been built upon some particular dataset borrowed from a scholar at CMU or elsewhere and originally created to carry out some rather specific study. These studies have ranged from a content analysis of the cahiers de doleances, or grievance lists, compiled on the eve of the French Revolution to a demographic profile of the workers in a Pittsburgh steel mill based on its surviving personnel records. In each case, someone has taken the raw data and created from them a manageable subset of working variables. The course staff then formulates, or asks the students to formulate, one or more plausible hypotheses which each student then tries to test by obtaining crosstabulations of relevant variables from a computer and writing an essay which analyzes them.

Certainly these exercises have been valuable for the students and for the faculty as well. They entail, however, some distinct problems. In each case someone, usually a faculty member, has put a great deal of effort into creating the exercise, and naturally one would want that effort to pay off for more than one semester. Unless the staff can muster heroic ingenuity in varying the assignment each semester, however, they may begin to see the same essays being recycled. Moreover, at CMU we like to keep our introductory courses alive and interesting to faculty as well as students by encouraging the faculty to revise the syllabus regularly. A computer exercise on a particular topic into which a colleague has put considerable effort can be an obstacle to this healthy spirit of innovation.

So alongside a desire to design a piece of software that most historians would see as facilitating their work in its most essential features, not simply on its margins, I also wanted to solve some specific practical pedagogical problems at the time I conceived this project. I wanted to create a general purpose utility which would access such a variety of data that we could do something different with it each semester as easily as we decide to substitute one paperback book on the syllabus for another. It was at about this time that I became aware of the capabilities of advanced function workstations such as those for which the ANDREW system was being designed by CMU and IBM. In my own research on agrarian violence in early nineteenth century Ireland I had done some statistical mapping using BASIC programs and the graphics capabilities of a PC. One of the things I had noticed was that even historians who are ordinarily not comfortable with statistical approaches were able to relate to the maps I was producing.

And so I hit upon the idea of building a utility which would display maps of the aggregate census data by county for the United States from the early nineteenth century to the present and the presidential election returns by county for the same period. Machine-readable files of these data are available from the Inter-University Consortium for Political and Social Research in Ann Arbor, Michigan. If such a program could enable the user to compare a map of one variable with that of another and/or with a map of physical and cultural features, then it would be relatively easy to design an almost limitless variety of class exercises. Students might be asked to explore the relation between ethnicity and voting patterns in the 1920s one semester and to investigate antebellum agricultural patterns and slaveholding the next semester.

Early in this endeavor I was put in touch with Tom Peters, an IBM systems designer who had been seconded to the ANDREW project. He was working on two subroutine libraries: GRITS, a set of database management facilities, and ZIP, which provided "zooming picture" capabilities. His interests and mine dovetailed almost perfectly. His subroutines made it possible for me to access just the sort of capabilities I needed from an ordinary C program, while my project offered an ideal problem domain within which he could test, extend and optimize his subroutine packages.

I decided to begin with the 1850 census. We had no difficulty creating a GRITS file containing 14 variables for the approximately 1600 counties which existed in that year. Creating a machine-readable county map of the U.S. in

1850 was more difficult. Although cartographic databases containing county boundaries do exist for the most recent census years, only fragments of the historical cartographic data have been digitized. Moreover, many cartographic databases are designed for high–resolution plotting at the standards of engineering draftsmen; our need was for highly simplified boundaries to maximize speed of display on a video screen in an interactive mode. It turned out that the Geography Department of the University of Maryland, Baltimore County, had recently published an atlas of county boundaries for each census year since 1840, all on a consistent scale and projection. I decided that our best alternative was to digitize the maps by hand from that atlas using a Houston Instruments HIPAD digitizer attached to an IBM XT and supported by BASIC programs written for this application. The operator of the digitizer is instructed to capture only the major vertices of each county –– in other words, not to try to trace every squiggle in the rivers.

The tasks connected with producing machine–readable maps have been easily the most time–consuming part of this project so far. The time spent actual operating of the digitizer to trace out the boundaries of all U.S. counties in 1850 was actually quite small, perhaps two or three person–weeks. However, it was necessary to write a program to transform the polygons representing counties into trapezoids, the geometric form required by the ANDREW window manager whenever an area on the screen is to be shaded. Moreover, there turned out to be many problems in making adjacent counties fit together exactly for a satifactory visual effect, and we had to spend a good deal of time editing the raw digitized data. (For some of this editing, by the way, we utilized CMUTUTOR, which is also being demonstrated at this conference by Bruce Sherwood.) As a result of this experience I have extensively revised our data–capture procedures, so that as we proceed this summer to digitize the changes in county boundaries from one census or election year to the next we should not have nearly the difficulties which we experienced with the 1850 map. I am optimistic that we can complete all the remaining maps in about the amount of time it has taken us to digitize and edit the 1850 map alone.

What I will be demonstrating in the poster sessions is a prototype utilizing fourteen variables calculated from the 1850 census. The user selects a variable from a scrollable list in an index panel similar to those which appear in ANDREW's mail system. A histogram of the variable appears in a legend pane, and in the main pane of the display a map of the US is shaded to reflect the values of the variable by county. The user has the capability to zoom in on a section of the map, to scroll it across the screen and to change the cutpoints, i.e. the values of the variables which define the intervals to be shaded differently. It is also possible to pop a region up into a pane covering the left-hand half of the main map and to display a second variable in another map of that same region in a parallel pane.

I have a programmer at work on a number of enhancements of this prototype. Perhaps the most important is an expression evaluator which will allow the user to calculate his own variables from the raw data. If the user wants to look at the ratio of cows to Lutherans, I say let him! We also need to modify our data structure slightly so that we can accommodate data for different years having differences in the county boundaries. For example, the user might well want to compare the percentage of foreign born in the 1850 population with the vote for, say, the Whig candidate in the 1852 election or to look at the same variable for several successive censuses. In addition, I am of course well aware that there are statistically appropriate ways to test hypotheses which one might develop from the inspection of maps such as these. These include ordinary regression techniques modified to accommodate spatial autocorrelation analysis and thereby take account of diffusion patterns and contagion effects. I shall be more than happy after this session to discuss this matter further with anyone who understood this last sentence. Let me just say now that we do intend to incorporate appropriate analytic capabilities into the utility, but that is not our first priority. Put slightly differently, we intend that it have research applications as well as educational ones.

Since we have not yet tested this utility in a course, I cannot report any findings on its educational effectiveness. I can, however, report considerable enthusiasm among my colleagues for what can be done with even this limited prototype. The kind of maps we can produce in a couple of minutes (and we hope to optimize this further), would have taken a draftsman weeks to produce by the methods which were used to compile similar examples in certain printed historical atlases. Thus we are really seeing here the possibility of genuine new discovery from data which have been lying around for decades without adequate technology to exploit them. It is the excitement of discovery on the part of my colleagues which I hope will be communicated to students in the specific exercises built upon this utility.

Let me show you some examples. Figure 1 is a map showing the percentage of foreign-born in the 1850 population. The counties with less than 0.7% of their population born outside the U.S. are shaded with the lightest pattern. The medium shading pattern represents 0.7% – 10.9% foreign born and the darkest is for over 10.9% foreign–born. If I had preferred some other set of cutpoints I could have reset them with a series of mouse clicks in the legend panel at the bottom. A number of interesting features emerge as we examine this map. For one thing it dramatizes one of many differences between North and South on the eve of the Civil War. Recent immigration had been overwelmingly to the North. Notice that the only parts of the South which are shaded dark are areas which had been added to the U.S. during the previous fifty years –– Florida, Texas, Louisiana and the Mississippi Gulf Coast –– which presumably had residents who were counted as foreign–born because they were born there prior to annexation. In the North we can see at least two patterns of settlement: urban settlement on the East coast and extensive rural settlement at the frontier in Minnesota, Wisconsin and northern Michigan. Once we have more data in the system the user will be able, for certain census years, to break out the components of these phenomena into different nationality groups.

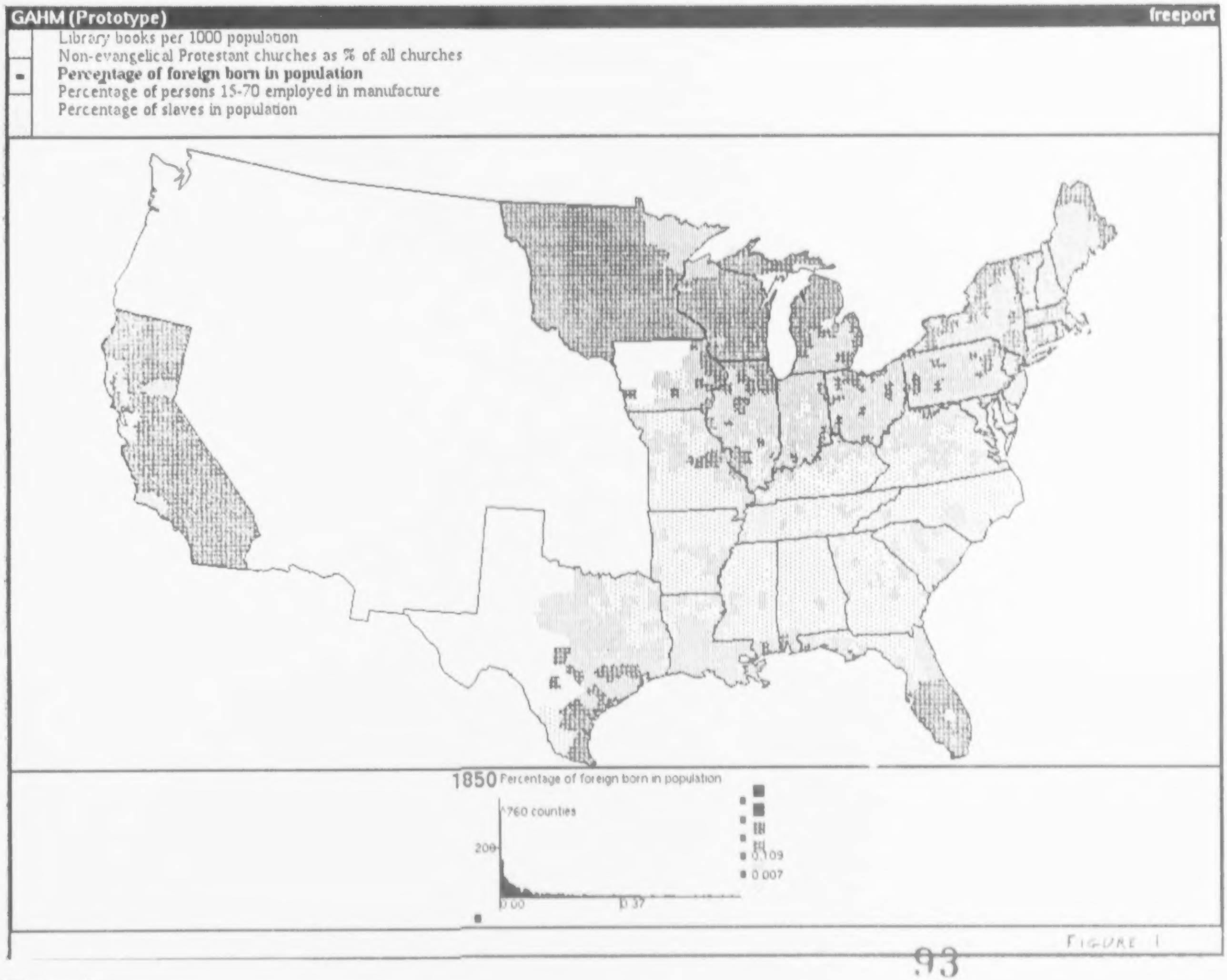
Now let me show you one region "popped up" into a detail pane (Figure 2). In the left pane you see a blow–up of the map you have just seen for Arkansas, Mississippi and Louisiana. Notice how such immigration as there was beyond southern Louisiana tended to follow the rivers. We could probably find examples of this phenomenon in the North also, though because of the generally higher level of foreign–born population we might have to use the capability to reset cutpoints in order to tease it out. To demonstrate the ability to compare maps in this pop–up mode, I have put the variable representing Roman Catholic churches as a percent of all churches in the right–hand pane.

Let me quickly show you some other maps which I hope will whet your appetite to come to one of my poster sessions. This (Figure 3) is the crude birth rate. Notice how it demonstrates that in the sections of the country which had been settled longest the population tended to be older and therefore the crude birth rate lower. High birthrates tended to be found on the frontier where the population was younger. Again, when we have more data in the system, and in particular the capability for the user to create his own variables, demographers using this facility will want to construct other measures of fertility which should then tell us different things.

In this map (Figure 4) of manufacturing employment, you see the capability to "zoom–in" on a section of the whole map. In this case I used the "reset cutpoints" capability to highlight those counties with exceptionally high employment in manufacturing. Apart from some freakish readings in the upper midwest which probably reflect very small populations, notice that almost all real industrialization is still concentrated on the east coast. Pittsburgh, which we think of as an archetypal industrial city, was still a predominantly commercial city in 1850 and its county falls into the same shading pattern as most other cities of the frontier.

Now let me conclude by reflecting on what I have just been doing. I have been picking out patterns on the maps by a process which draws upon some specific historical knowledge, some conceptual training, and perhaps some intuition as well. It will be important to include well-designed maps of physical and cultural features such as rivers, mountains, railroads and cities and certain sorts of written material in the exercises we design for students who do not yet have some of the information on which I am drawing when I talk about these maps. Provided we are careful to make explicit these sorts of information, however, I think we can use this tool to stimulate in students the same kind of pattern recognition which I have been demonstrating.

Although historians are perhaps better equipped than other people to make sense of such maps as these for 1850, members of many other professions would be able to make sense of similar maps for recent periods, perhaps in ways which would not occur to an historian. Moreover, the easily available data for recent times are incomparably richer than what we can obtain for the nineteenth century. A version of this utility accessing suites of data on disease patterns, for example, would I think be of great interest to a medical audience. A version which drew upon, say, a database of election returns, both federal and state, for the period since 1945 would interest not only political scientists but also professional politicians and journalists. A suite of data relevant to marketing issues could be designed with business users in mind. Each of these professional groups would bring to the data different sorts of questions and different specific background information which would affect the way they would seek and find patterns in the evidence. But in the fundamental activity of seeking patterns they would be behaving as historians whether they knew it or not.



GAHM (Prototype)

freeport

Library books per 1000 population
Non-evangelical Protestant churches as % of all churches
Percentage of foreign born in population
Percentage of persons 15-70 employed in manufacture
Percentage of slaves in population

FIGURE 1

Figure 1.

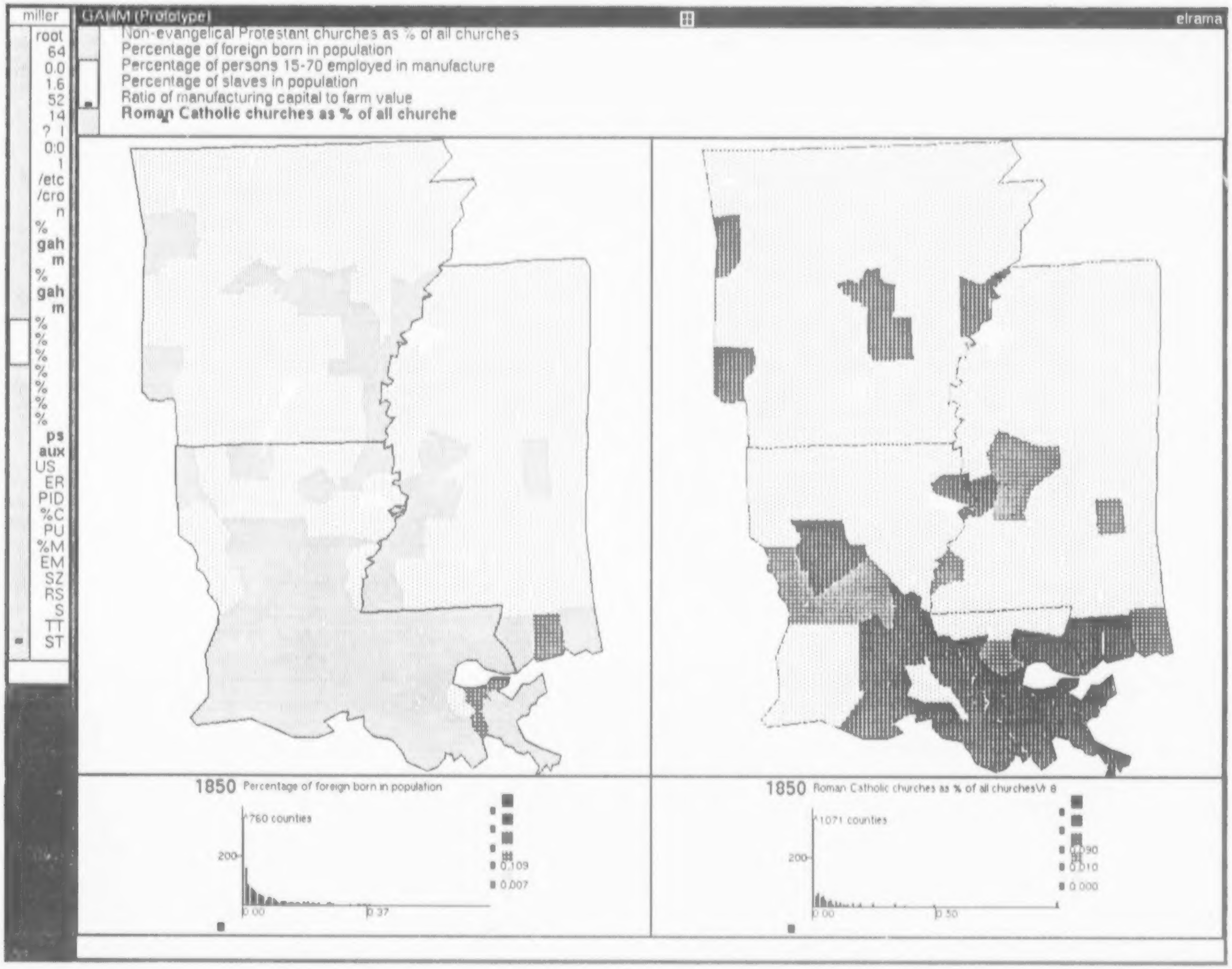

miller
GAHM (Prototype)
elrama
Non-evangelical Protestant churches as % of all churches
Percentage of foreign born in population
Percentage of persons 15-70 employed in manufacture
Percentage of slaves in population
Ratio of manufacturing capital to farm value
Roman Catholic churches as % of all churche
root
64
0.0
1.6
52
14
/etc
/cro
gah
gah
ps
aux
US
ER
PID
%C
PU
%M
EM
SZ
RS
S
TT
ST
1850 Percentage of foreign born in population
760 counties
200
0.109
0.007
0.00
0.37
1850 Roman Catholic churches as % of all churches\r 8
1071 counties
200
0.090
0.010
0.000
0.00
0.50

Figure 2.

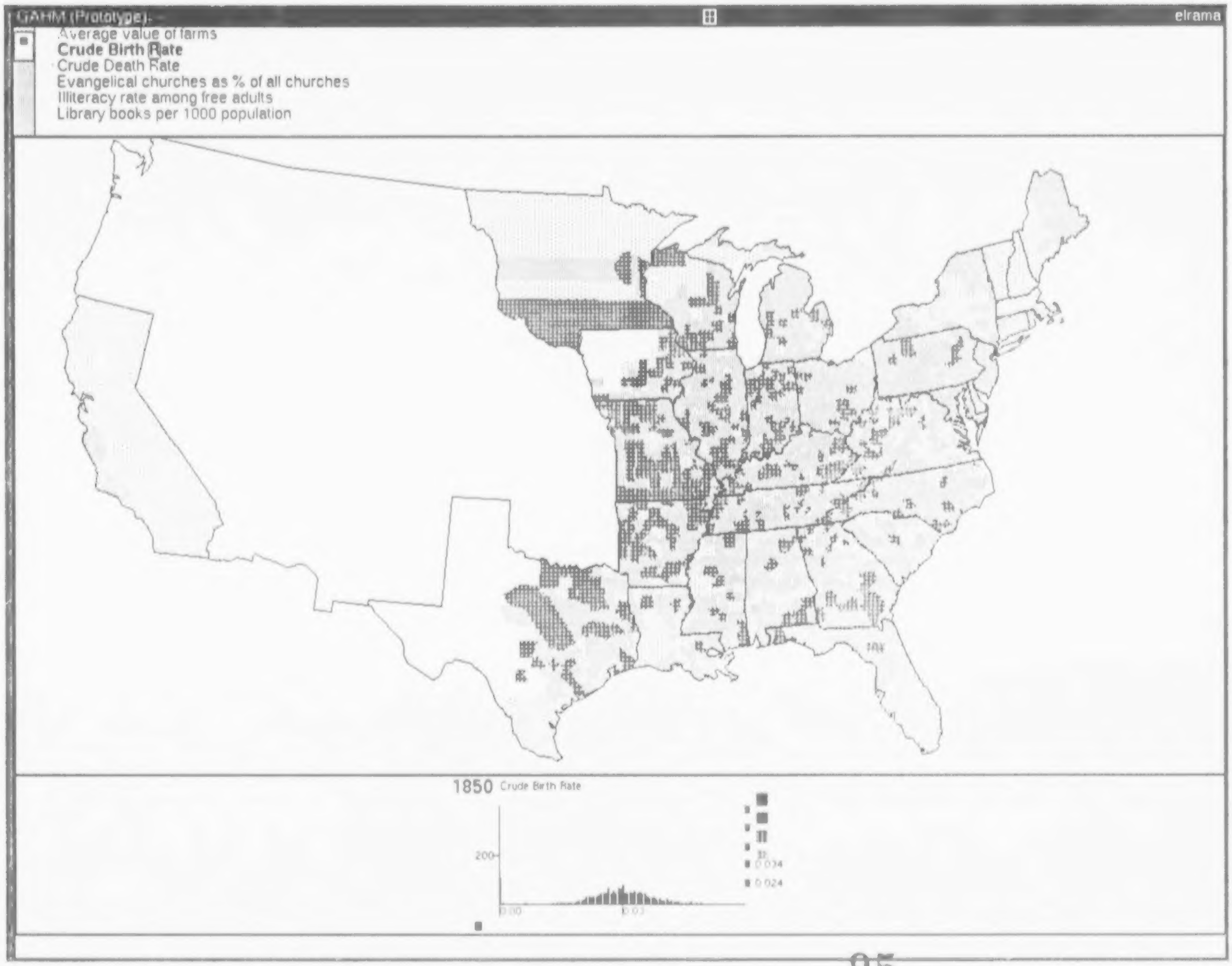

GAHM (Prototype)
elrama
Average value of farms
Crude Birth Rate
Crude Death Rate
Evangelical churches as % of all churches
Illiteracy rate among free adults
Library books per 1000 population
1850
Crude Birth Rate
200
0.00
0.03
0.034
0.024

Figure 3.

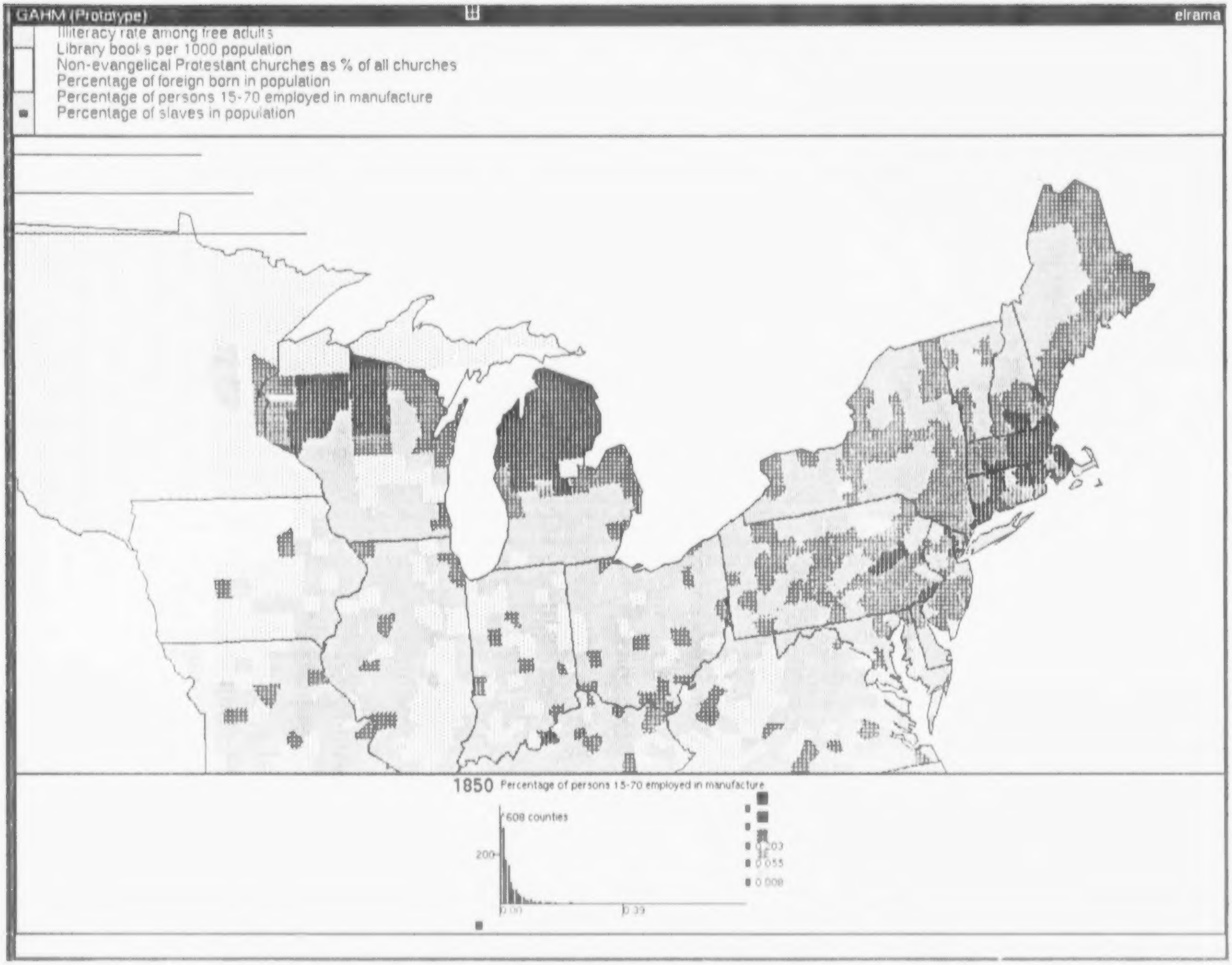

GAHM (Prototype)
elrama
Illiteracy rate among free adults
Library books per 1000 population
Non-evangelical Protestant churches as % of all churches
Percentage of foreign born in population
Percentage of persons 15-70 employed in manufacture
Percentage of slaves in population
1850 Percentage of persons 15-70 employed in manufacture
608 counties
200
0.00
0.39
0.203
0.055
0.008

Figure 4.

THE INS AND OUTS OF MACROECONOMIC FORECASTING: A TUTORIAL FOR STUDENTS

Presenter: John Haltiwanger *University of California, Los Angeles*

Students who go onto careers involving economics in business, government, or academia are likely to be potential purchasers of and users of macroeconomic forecasting services. Underlying the forecasting services is the forecasting model which consumers typically perceive as a blackbox. The software development and the associated computer exercises from this project are intended to provide some perspective on the type of macroeconomic forecasting models that are available and the pitfalls involved in forecasting. The goal is to make intelligent consumers out of students who have an intermediate undergraduate macroeconomics background and have limited familiarity with both statistics and computers.

The software and exercises that have been developed provide the student with the type of output forthcoming from macroeconomic simulation models and through an interactive program allow the student to manipulate the models. After guiding the student through several alternative macro simulation problems, the student is asked a series of questions to interpret the alternative simulations. A key part of this is to require the student to analyze the output through a set of written exercises. One lesson to be learned here is that the computer and the software are powerful tools but it requires considerable effort and thought to interpret and use the output.

The alternative simulation problems and associated written exercises introduce the student to the notions of static and dynamic simulations as methods for gauging how good a model is. Moreover, the simulation problems and questions introduce, without requiring much statistical background, the notion of forecast errors and confidence intervals. In addition, while the software and exercises are still under development, they have been designed to allow students to make comparisons across different types of macroeconomic forecasting models that are available. The goal is to include for purposes of comparison a structural IS–LM type model, a reduced form monetarist model, a times series model, and an equilibrium business cycle model. By incorporating all of these models, the student will become familiar with the alternative models available and their different implications. Moreover, since the course material in the typical intermediate undergraduate macroeconomics courses involves presenting the theoretical underpinnings for each of these alternative macroeconomic models, this links the computer exercises to the course material in an integral fashion.

In the next section of this paper, we briefly summarize the overall computing strategy. In section III, we discuss the the results achieved to date. Finally, in section IV, we present some concluding remarks including discussion of plans for further development.

II. DESIGN AND IMPLEMENTATION

The software and associated exercises are intended for students enrolled in an intermediate level undergraduate course in macroeconomics. Students are assumed to have some familiarity with macroeconomics, limited familiarity with statistics, but no familiarity with computers. The objective is to provide an environment in which a student can insert the floppy disks into the PC, turn the PC on, and then have the software guide the student in an interactive manner through the exercises.

To accomplish the objective, the primary program is written in "C" and provides an interactive environment for the student to analyze alternative macroeconomic forecasting models. Once the system is turned on, the student is booted into the Macroeconomic Model Simulator program. This program first informs the user that they have control over several aspects of the analysis. Specifically, they can choose which of several popular macroeconomic models to use and which exogenous variables to change. Having made these selections, the student is asked what period of time to analyze and what magnitudes for the exogenous variables (e.g., the growth rate of the money supply) are to be used. The program then takes the students choice of model, time period, and exogenous variable changes and simulates the chosen model appropriately. After completing the simulation, the program asks the student which variables from the model simulation they would like to see (e.g., GNP, Inflation, Unemployment, Interest Rates). Students can either see the results in tables or in graphs. This is done in a sequential fashion. First, students are given the opportunity to observe tables indicating the predictions (and the actual numbers if the forecast is done over a period in the past). Then, the students are automatically booted into LOTUS where the output awaits them for graphics. Macros have been written for LOTUS which accept the output and then in an interactive fashion ask the student what output to graph. Included in the output are both the forecasts and forecast errors. The graphs provide bands indicating appropriate confidence intervals for the forecasts.

This shell which has been written in "C" allows for an extensive set of exercises. It is written in such a way that the novice can get into little trouble. Incorrect responses to queries invoke a polite error message and instructions as to how to provide an appropriate response. At the same time, the student has control over the key aspects of the analysis.

Accompanying the software are extensive sets of written instructions and accompanying exercises that they are to complete with this program. To date, these exercises include dynamic simulation of a model over a period in the past to demonstrate how most models tend to wander off course eventually, out-of-sample forecasting exercises, and forecasting exercises using dramatic policy variable changes to demonstrate how poorly models tend to do with policy changes that are out of the range of the period for which the model is estimated. The software is flexible so that additional exercises could be incorporated.

III. RESULTS

The current version of software and exercises has been successfully used in three quarters of an intermediate macroeconomics class with an average enrollment of 175 students. Students have been provided with an extensive hand-out outlining the assignment and teaching assistants provided instructions in class sections. Students have been given two weeks to complete the assignment. Each student was required to go to the micro computer lab (which has approximately 50 PC's), check out the software, and complete the assignment on his/her own. Two floppy disks are checked out by the student: the LOTUS disk and the program disk. Students are instructed to insert the LOTUS disk into the A: drive and the program disk into the B: drive and then turn the machine on. Students are provided instructions on how to exit LOTUS and access the program. After this, the software is designed as indicated above to guide the students through the exercises. Teaching assistants who staff the computer lab were also given extensive instruction on how to use the program.

For the most part, students with no experience with computers have been able to complete the assignment successfully. We have found that a key element to guarantee smooth handling of the assignment is to maintain good communication with the teaching assistants staffing the computer lab. The information they conveyed to us based upon their actual observation and monitoring of the students enabled us both to make modifications improving the software and to provide helpful hints to students during class hours to avoid common errors that the teaching assistants had observed. Overall, although the software is still under development, we have demonstrated that assignments of this type can be completed quite successfully with large undergraduate classes in a decentralized manner even with students with little computer experience.

IV. CONCLUDING REMARKS

Many of the students in economics are likely to be users of macroeconomic forecasting models at some point in their careers. Purveyors of these models often provide visually impressive output. Who among us is not impressed with colorful graphics depicting predictions of key economic variables based upon seemingly sophisticated econometrics? However, both macroeconomics and forecasting are still in the developmental stage. There is considerable disagreement within the economics profession about the appropriate model and this makes the process of forecasting quite difficult. The software and the associated exercises developed in this project are intended to provide students with a healthy degree of skepticism with regard to macroeconomic forecasting. We make it clear that decision makers at almost all levels (government, business, consumer) require economic forecasts but that one needs to recognize as well the pitfalls involved in making forecasts. We attempt to provide this skepticism in a non–technical manner so that not only the sophisticated have some perspective on these issues.

This project is still under development. Not all of the macroeconomic models that we would like to include for purposes of comparison have yet been incorporated. Further, students are not permitted to modify the specification of the models under the current arrangement. We intend to allow students to modify the specification and to design some exercises showing how key aspects of the model specification are fundamental for the results that are derived.

INDEX BY PRESENTER / AUTHOR, VOLUME I

Presenter / Author	Page
Fliess, Kenneth H.	52
Goldman, Steven M.	2
Gutmann, Myron P.	52
Haltiwanger, John	108
Holmes, Randy D.	45
Kidd, Kenneth K.	45
Kroch, Eugene	85
Lachenbruch, Peter A.	62
Lee, James L.	13
Miller, David W.	97
Miller, Kevin F.	34
Rae, Douglas W.	66
Rohn, Peter H.	112
Rosenstone, Steven J.	66
Shapiro, Matthew D.	40
Stolp, Chandler	23
Williams, David R.	28

INDEX BY UNIVERSITY, VOLUME I

University **Page**

Carnegie–Mellon University 97

University of California, Berkeley 2

University of California, Los Angeles 62
108

University of Pennsylvania 28
85

University of Texas 23

University of Texas at Austin 34
52

University of Washington 112

University of Wisconsin, Madison 13

Yale University 40
45
66

IBM

Academic Information
Systems Proceedings

1986 University AEP Conference

Volume II:

HUMANITIES AND ARTS / DESIGN

April 5 – 8, 1986
Sheraton Harbor Island Hotel
San Diego, California

CONTENTS, VOLUME II

Title **Page**

A Computerized Sumerian Dictionary of the Third Millennium 2

A "Language Lab: Design for Architectural Design ....

An Authoring System as a Tool for Foreign Language Learning ....

Bilingual Word Processing: A New Tool for Learning ....

Computer Text Analysis/Database Management, Musicological Study 45

Interactive Videodisk and Foreign Language Learning ....

Musical Acoustics Demonstrations/Lessons 60

Philosophical Simulations and Experiments: Scientific Reasoning 64

Remote Electronic Music Synthesis Workstations 78

Report on Logic: A Workbook ....

Software for Printing South Asian Scripts 91

The Cairo Geniza Project 101

Workstations for Computer Music at Carnegie-Mellon 109

INDEX by Presenter/Author, Volume II 118

INDEX by University, Volume II 119

A COMPUTERIZED SUMERIAN DICTIONARY OF THE THIRD MILLENNIUM

Presenter: David I. Owen *Cornell University*

copyright © David I. Owen, 1986

Consider for a moment a corpus of documents numbering close to 100,000 and containing millions of lines of text written in a partly ideographic and partly syllabic cuneiform script much of it polyphonic (somewhat like modern Japanese). Consider also that this corpus consists of ca. 50,000 published texts which are mostly in facsimiles, some in transliterations, and a small percentage actually edited and translated. Now picture this corpus as written on varying sized and shaped clay tablets, from 2 cm to 30 cm high, scattered in museums and universities around the world and inscribed with mankind's first written language. After a century of sustained study, the language still lacks a comprehensive dictionary and even an up–to–date grammar. Such a picture might be the nightmare of some scholar, but in reality it is the general state of affairs with respect to the study of the Sumerian texts of the third millennium B.C.

For much of its history, the analysis of these texts has been in the hands of a small group of dedicated scholars who simply created personal card files based on either published sources or on their own studies and translations of texts. To some extent the situation changed dramatically last year with the publication of the first volume of Pennsylvania Sumerian Dictionary at the University of Pennsylvania consisting of words beginning with the phoneme B. In another two years we are promised the second volume, to be published in two parts, of words beginning with the phoneme A. The book catalogues have announced the completion of this dictionary for the year 2022. But let me point out that the director of the project upon whose personal files the dictionary is based, is now approaching sixty. The dictionary staff itself is made up of only three additional members two of whom are maintained on soft money. With this not atypical scenario it is no wonder that progress in the study of the vast Sumerian resources for the third millennium B.C. has remained rather static and limited to a very few individuals.

The situation began to change in the early seventies with the the establishment in 1972 of a Sumerian Dictionary Project at the University of Rome under the direction of G. Pettinato. The goal of this project was to compile materials for a

Sumerian dictionary limited to the corpus of texts from the third millennium B.C. Shortly after this project began, an expedition of the University of Rome discovered a vast library dating to the twenty–sixth century B.C. at the Syrian site of Tell Mardikh, now known to have been ancient Ebla. Following quickly upon the unearthing of some 15–20,000 whole and fragmentary tablets, some containing over 3000 lines each, the texts were recognized as having been written in Sumerian and in a hitherto unknown Semitic language which the decipherer, Prof. Giovanni Pettinato of the University of Rome, designated as Eblaite after the ancient name of the city. In addition to the mostly economic texts in the library, there were also literary texts, international treaties, mathematical texts, magical and religious texts and, most remarkably of all, dictionaries containing up to 1500 bilingual entries in Sumerian and Eblaite. As he began to prepare these unique documents for publication, Pettinato realized that this mass of new data on Sumerian, coupled with the new materials on early Semitic, could not be adequately studied or absorbed without some kind of reference dictionary which incorporated both the old and the new sources. Given the huge number of texts already available for study and the coincidental expansion of computer technology in the late seventies, he decided to try and organize a cooperative venture among scholars to create an electronic database which would contain the corpus of texts, published and unpublished, in transliterated and machine readable form. Once this corpus was available in such a database, it could be easily sorted and studied, updated and corrected, and utilized to create both computer generated, specialized, publications as well as an on–line, encyclopedic dictionary. Furthermore, it would mean that the enormous task of transliterating and translating all these texts would have to be done only once and that current and future generations would have at their disposal this great resource to expand upon as new texts became available and as our understanding of the languages improved.

At the same time Sumerian dictionary projects were also being organized with computers at the universities of Freiburg and Berlin. During an international meeting held at Freiburg in 1980 to discuss the prospects for a Sumerian dictionary, it was decided to combine efforts under the single auspices of the University of Rome to be directed by Pettinato. The University of Rome quickly grasped the implications of such a project and the Computer Center, under the direction of Prof. Mirella Schaerf, joined in the effort by creating a Fortran program for the database on their Sperry mainframe. The Sumerian Dictionary Project was elevated to the status of a University Project, the highest priority designation for research in the humanities faculty. Staff was hired to begin the input and coding of data based on the newly developed mainframe program. Discussions with several universities in Germany began in order to divide up the material for study and coding. To date the universities of Berlin, Freiburg, and Heidelberg are working on the project and Cornell has joined as the first American university (1). Needless to say the progress has not been dramatic given the pace of transliteration that is required during the input and coding stages coupled with the fact that there is no full time staff. Since most texts are read directly from original tablets, facsimiles, or entered from transliterations, the work must be done by specially qualified individuals. Once the texts are typed into the database each line must be coded so that it may be utilized by the Dictionary Program for sorting and other analyses. The latter process is a bit faster since less skilled individuals may be utilized at this stage.

It may be helpful at this point to detail some specific features of the Sumerian language which make this project so unusual. Sumerian is not related to any known language, ancient or modern. It is unique by virtue of its structure and by the fact it is the world's oldest written language. From a structural point of view, the language is of the agglutinative type and has been compared to Basque, Hungarian, Turkish and Finnish. In agglutinative languages grammatical and syntactical relationships are expressed by the addition of grammatical modifiers to an immutable root. For instance, in Latin the word pueros contains the ending –os which signifies that it is masculine, plural and in the accusative. In Sumerian, however, each of these concepts would be indicated by a different grammatical morpheme, one for each and attached sequentially to an unchangeable root or base.

In addition, the writing system used to express Sumerian is both polyphonic and homophonic. That is, a single cuneiform sign may have more than one reading and meaning; or different signs may have the same reading or sound but different meanings. Over the years scholars have organized a universally accepted reference system which assigns to each cuneiform sign Latinized characters of different numerical values (e.g. gi, gi2, gi3, gi4, etc.).

Another characteristic of Sumerian is its ability to refer back from within the verbal chain to some part of the preceding nominal phrase. Thus it is essential for any computerized dictionary of Sumerian to be able to distinguish between a real lexeme and a simple grapheme. A single example should suffice.

den–lil2–la2 ur–dnamma lugal–uri2ki–ma–ke4 e2–mah–ha–ni –na–ab–du3–e "For Enlil, Ur–Namma, king of Urim, his lofty temple for him (i.e. Enlil) he will build it."

The first element, the god dEnlil, is formed by three graphemes to which is added the dative indicator –ra which, because of Sumerian phonetic rules, becomes –la2. Ur–dNamma, the name of a king, is also formed by three graphemes; then the affixed "king of Urim" formed by the lexeme lugal (king) and Uriki–m (city name) plus –ak, the mark of the genitive, followed by –e, the ergative or subject marker; this is followed by the direct object e2 (house, temple), followed by the adjective mah (lofty) and the pronominal masculine third person singular possessive pronoun; followed by the third person singular transitive verb du3 in the present–future.

The grammatical transcription of the line would read as follows:

den–lil2–ra ur–dnamma lugal–urim2ki–ak–e e2–mah–ani mu–na–b–du3–e

In order for the computer to search for specific lexemes it is necessary to designate, in some way, which elements are, in fact, lexemes. Of the phrase provided, only lugal, e2–mah and du3 will be made part of the dictionary since all other elements will be assigned to categories other than lexemes (= entity endowed with its own meaning). This then brings us to the procedure followed by the computerized dictionary which, in brief, can be described as follows.

1. Transliteration of original texts
2. data input
3. codification of data by lexemes and by graphemes
4. processing (separation of lexemes into archives and syllables into a syllabary)
5. Computer generated, printed volumes

Examples of each stage are in order here. It should be stated at the outset that the basic underlying concept of the codification is to link a coded label to each text to identify it by sign, number, provenance, type and period and to attach to each line of text strings of code that represent the lexeme and grapheme subdivisions which determine to which typological class each belongs. The latter would include, personal names, names of deities, administrative terms, agricultural and industrial terms, verbal forms, etc. In addition, the codification of the graphemes links each one to the original cuneiform sign(s). These are preceded and followed by dashes whereas the lexemes and their typologies are designated by letters of the alphabet. For example, the phrase ki–i3–ka–la–ta, "on behalf of Ikalla," would be coded in the following way: the lexeme i3–ka–la has the code mark "N" which designates it as a personal name. The element ki– ta (two graphemes which precede and follow the personal name) is coded "a" which designates it as an administrative term. Thus the line of text will be coded –a3N–a. The "–" is used to unite syllables (those forming a lexeme) but which are not in sequence. The numeral before the lexeme designation "N" shows where the lexeme begins and how many syllable signs/graphs it contains.

In those instances where a lexeme is composed of parts of differing typological categories, additional codes are entered to to clarify these designations. These are entered in parentheses. For example, gu4–apin, "plow–ox," is made up of the lexeme gu4, "ox, bovine," plus apin, "plow". The phrase would be assigned to terms relating to agriculture with the code "A" and placed in the archive of agricultural texts and "2B" for the the terms relative to large animals. The form of the code would then be A(2B). Within each typological archive the terms would be arranged in alphabetical order.

One further designation is worth pointing out here. It is often the case in Sumerian where it is necessary to separate two distinct elements of a single transcription. This is done through the use of the equals sign (=). For example, gar–ra, "to put," would be rendered gar–r=a and encoded as V(rHx) where V desig– nates a verb, r, a root, H, an extra or unnecessary syllable or phoneme, and x, a verbal suffix.

For the past five years, work on the Sumerian Dictionary has been concentrated on the transliteration and encoding of texts from the Third Dynasty of Ur (21st century B.C.) and of the texts from Ebla (26–25th centuries B.C.), in total ca. 20,000 tablets. Furthermore, these data have been sorted into preliminary reference files of lexemes. These sort programs will allow scholars to consult the data so far encoded in ways hitherto impossible given the huge number of texts. Documents may now be sorted by date, by provenience, etc. or lists of personal names, place names, or administrative terms can be extracted and localized by date, place, etc. Furthermore as more texts have been added to the database, the program has been expanded to include more facilities for manipulating the data.

Once the principles and guidelines were established for each typological file, a complete listing of lexemes in each archival file was prepared along with a corresponding list codification. These two lists will form the basic reference tool for subsequent work on the dictionary and will be provided to scholars working on the project along with specific instructions for classification and codification. For the present, it was decided to publish a computer generated sample volume to illustrate the methods employed in our codification and the results which can be obtained through the use of the database as presently compiled. The volume is addressed both to Sumerologists and to those interested in the application of computer technology to dictionary compilation and linguistic analysis. One cuneiform text volume, published in 1901 but never completely transliterated or translated (G. Reisner, Tempelurkunden aus Telloh), was selected for analysis. The first Dictionary volume appeared in November, 1985 in three unequal parts: I. Transcription of the Texts: II. Typological Archives: III. Glossary of Terms, in total over 1500 pages of computer generated text (2). This publication will serve as a vehicle for discussion with a wider group of scholars whose criticisms and suggestions for corrections and improvements will be utilized in future work on the database and publications.

Through the research conducted on the compilation of a program for lexical selection in a language of agglutinative structure like Sumerian, we have learned that the process is much more simple than that with an inflected language. The correspondence between formative combinations and grammatical and syntactic relations in an agglutinative language is quite rigid. Whereas in an inflected language even the isolation of a lexical root is quite difficult for a data processor. But, whereas so much work has already been completed with respect to computer applications to inflected languages, little, if any, has been done with agglutinative languages. Thus the development of programs for our dictionary should have much wider application than might be apparent at first glance. The fact that Sumerian has rigid syntactic relations expressed by modifiers which are almost invariable in form, makes the analysis of its linguistic structure easier given the logical strings of commands used by most language compilers. Even recognizing the differences between the subjects with which we are dealing, the computer program adopted for our dictionary works very much like a compiler. Thus, if we can demonstrate that our method of coding can be used effectively as a tool for grammatical study, we will have succeeded in making this effort useful for the study of other languages, both ancient and modern, of the same basic structure.

In closing, it may be of use to summarize the composition of the database (3). Its organization allows all texts be retrieved by graphemes or by lexemes with which they are composed, but will also allow the texts to be retrieved according to attributes which are organized by time and space: period, probable or actual date (year, month and day), provenance, classification. The database thus functions as an "electronically controlled dictionary" which is organized into five primary sections (figs. 1–2) (4):

a) Archive of texts
b) Archive of codes
c) Archive of attributes
d) Complete glossary (translations into four Western languages)
e) Typological files

The Archive of Texts will allow for the creation of the "Sumerian Dictionary of the Third Millennium" and will form the basis for the study of the evolution of the language over a millennium. The Typological Files will permit the unified codification of all future texts. And with the completion of the database we shall be able to study the use of syllables, phonemes and graphemes; the composition of lexemes and syntactic units; word function and organization in differing categories; and affinities and reciprocal relationships between categories (in essence, morphological analysis). This has already resulted in the arrangement of the data into

1. Complete texts
2. Lexical and Sign Codes
3. Syntactic units within a text
4. Lexical units forming syntactic units
5. Syllabic elements comprising lexical units
6. Glossary

In addition, although a line of text contains a keyword sufficient for the listing of such words in separate typological files, this is not enough for the dictionary; but it makes the entire syntactic unit indispensable. These keywords are therefore placed into "inverted" type files each one of which is associated with a list of all texts within which the keyword is found. In practice this means that the label of each document is itemized and contains all the information necessary to separate the text, the syntactic unit, the line of text, or the lexical unit itself.

In bringing this perhaps overly technical description to a close, let me note that the Sumerian Dictionary Project portends to be a major step forward in the study of the civilization of the most ancient Near East. It will, for the first time, make accessible the extant written corpus of documentation from Syria and

Mesopotamia for the entire third millennium B.C. It will allow the scholar and student to study this data as never before and to extract information hitherto impossible. Furthermore, it should enhance our understanding of the Sumerian language and its two other contemporary languages of Akkadian and Eblaite. This huge corpus of documentation is but a remnant of a far greater written body of material which has been wantonly destroyed or accidentally perished. We are, perhaps, to be compared to the Babylonian scribes who recopied and preserved the Sumerian texts, the Masoretes who reproduced and vocalized the Hebrew bible to preserve it, and the Medieval monks who labored to copy and preserve the literary remnants of Graeco-Roman civilization for posterity. We humanists today, working together with computer scientists, are attempting no less by continuing in this tradition. But today's possibilities are far more exciting and promising. I for one am particularly pleased to be playing a small part in this process by trying to preserve the world's oldest written records by way of the world's most advanced technology. It is, indeed, among the highest ideals of the University that such research is encouraged and supported in cooperation with industry. The legacy of the past, through us, will be our legacy to the future.

ACKNOWLEDGEMENT

I wish to acknowledge my thanks to my colleagues at the University of Rome, Profs. Giovanni Pettinato and Mirella Schaerf, who discussed with me many of the problems alluded to in this paper and provided me with extensive documentation for the statements made below. This paper is substantially in the form it was presented in San Diego with the addition of footnotes and minor additions and changes.

FOOTNOTES

1 For a more detailed history of the Sumerian Dictionary Project, see the introduction to G. Pettinato, et al., Studi per il Vocabolario Sumerico 1/1, Roma, 1985, v–xviii.

2 Studi per il Vocabolario Sumerico 1/1–3: G. Reisner, Tempelurkunden aus Telloh, Roma, 1985.

3 For a more detailed account of the organization of the database, see Guiseppe Visicato, "Una metodologia Informatica per la Realizzazione di un Vocabolario Sumerico del III Millennio," in Oriens Antiquus 21 (1982), 111–122.

4. Adapted from G. Visicato, op. cit., p.115 and 117.

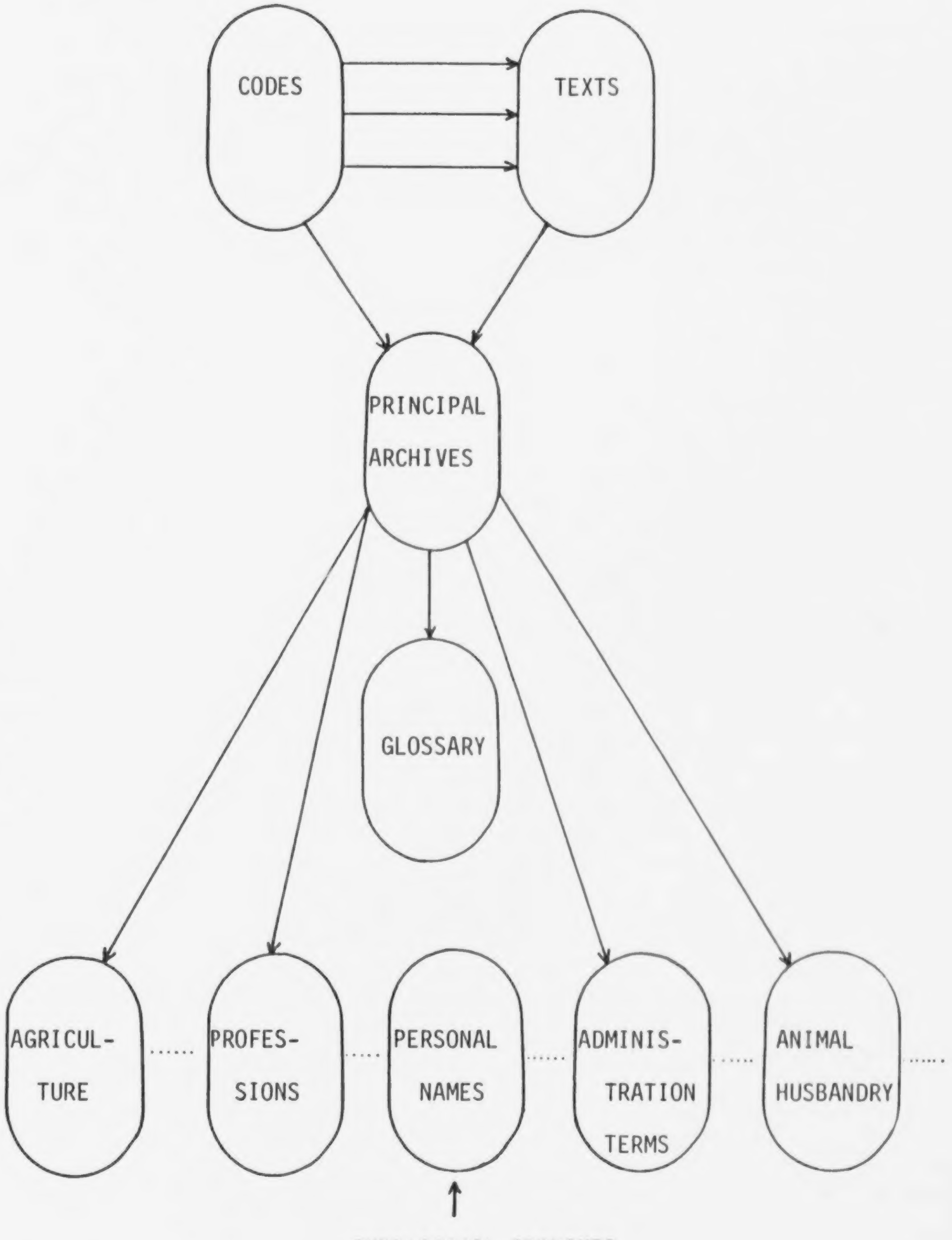

CODES
TEXTS
PRINCIPAL ARCHIVES
GLOSSARY
AGRICUL-TURE
PROFES-SIONS
PERSONAL NAMES
ADMINIS-TRATION TERMS
ANIMAL HUSBANDRY
TYPOLOGICAL ARCHIVES

Figure 1.

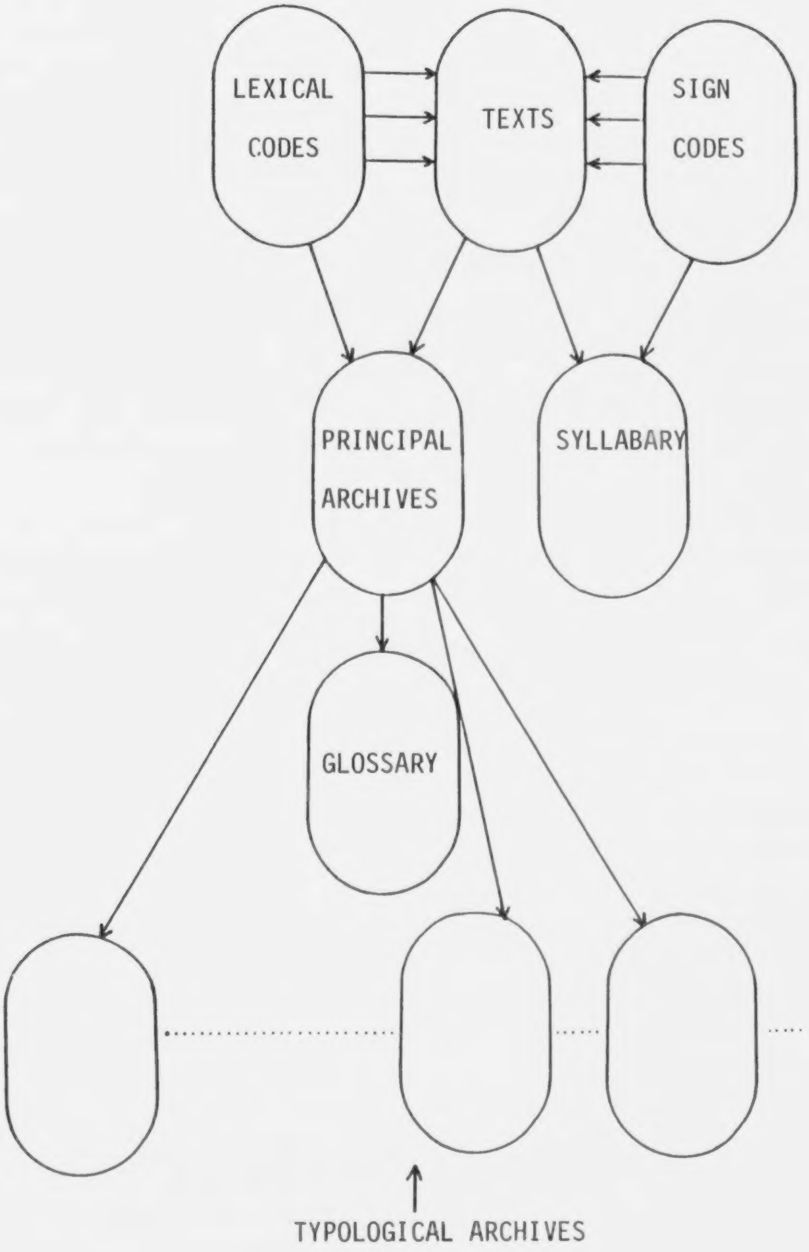

LEXICAL CODES
TEXTS
SIGN CODES
PRINCIPAL ARCHIVES
SYLLABARY
GLOSSARY
TYPOLOGICAL ARCHIVES

Figure 2.

COMPUTERIZED TEXT ANALYSIS AND DATABASE MANAGEMENT AS TECHNIQUES IN MUSICOLOGICAL ADVANCED STUDY

Presenter: John W. Hill *University of Illinois*

copyright © John Walter Hill, 1986

Under Project EXCEL, the musicology division of the School of Music was awarded six IBM ATs, which are now located in faculty offices and common rooms on the fourth floor of the Music Building, where faculty and most graduate students in musicology have offices. These computers are and will be centers for project development before, during, and after seminars that engage faculty and students in mutually shared research and learning. The seminar being offered this spring and all the others currently planned make use of databases and computerized text analysis.

The first of these seminars, "The Italian Contrafactum from the Renaissance to the Eighteenth Century," currently involves my students and me in the development of three text databases with similar purposes. Each of these databases has as its purpose to identify texts (lyrics) that are known to us in purely literary collections, without music, and to match those lyrics to the melodies to which they were once sung. The music to which these lyrics were sung will be found with entirely different words because composers, musicians, and amateurs of the past (like many today) reused old songs for new purposes by substituting new lyrics for the original texts. In most cases, however, only one set of words is preserved with the music; the other lyrics, as I have said, will be found in purely literary collections of poetry. But there are clues in the form and content of these tuneless lyrics that help us to provide them with their missing melodies. The clues emerge when we find specific types of similarities between the original words to the songs and the new lyrics.

My personal project formed the model and starting point for the spring seminar. My database contains all the texts (about 3,000 in number) to all the arias from all productions of all the operas composed by Antonio Vivaldi, one of the most prolific opera composers in history. Previous study of the surviving scores to Vivaldi's operas had revealed that he, like most of his contemporaries, frequently reused arias from earlier operas. In doing so, he normally fit new words to the old music in order to make the lyrics fit the new character and situation. The nature and extent of this self–borrowing, however, was not fully known because scores survive for only about one–third of Vivaldi's operas. For the others, we have only the word books (libretti). But study of the borrowing and retexting revealed to us in Vivaldi's surviving scores allows us to extrapolate to his operas known only from libretti. Using processing, transformation, comparison, storage, sorting, retrieving, importation, exportation, and split–screen display tasks that I programmed using the Savvy PC relational database system language, I have been able to identify and match up Vivaldi's texts that were sung to the same music and to store groups of such matches for easy retrieval and display. The results have been very significant.

The main purpose of this text analysis is to gain insight into Vivaldi's working principals of music–text relationship. You see, it was a fundamental tenant of late Baroque aesthetics that the music of an opera aria should express the same passions that the words and the drama (plot, character) implied. But some modern scholars have challenged the validity of that premise by pointing out that even such composers as Bach and Handel frequently fit new text to old music. How could the same music express the passions implied by often very different texts? What does Baroque theory have to do with the reality of composers' practices?

One of the results of my aria project has been to show that Vivaldi did have very clear and interesting principals in mind when fitting new words to old music. If the specific passion was not retained in the new text, then at least the new text would imply a passion with similar characteristics of (in Baroque theoretical terms) relative amount of expansion of the life spirits, activity or passivity, or pain versus pleasure, for example. Also certain principals of text declamation were maintained along with features of vowel sound, poetic accent and meter, etc. Understanding Vivaldi's working principals of music–text relation helps us to interpret and experience his (and his contemporaries') operas in terms of appropriate to the composer's intentions and the mind–set of his audience.

Among the other results are a more secure dating of surviving music and some indications that Vivaldi himself may have been the poet who wrote the new texts for his old arias.

One group of students in my spring seminar is working on discovering music for sixteenth–century Italiam stage works. They are finding the music hiding in madrigal collections, disguised by being reclothed in a generic text with the original topical references removed. The strategies and programming routines that these students are using are very similar to those I used in the aria index, but the repertoire and objectives are quite different.

Another group of students is working to find the music of fifteenth-century popular Italian songs to which the words of spiritual hymns (laude) were sung.

Future projects along these lines will seek to discover the original texts for some arias, ensambles, and choruses in Bach's church cantatas.

Meanwhile one colleague, Tom Ward, is preparing for a seminar, to be offered next year, in which the texts of medievil music treatises will be compared, with the objective of finding paraphrased borrowing. Some of the programming tasks developed in my seminar will be modified for use in his. And some of the same students will participate in both.

Later, Professor Lawrence Gushee will use the same database programming language to develop a computer-based index of biographical and demographical data on early jazz musicians in New Orleans. He will apply some of the same techniques and methods to arrive at a composite biography of medievil French minstrels. Professor David Stigberg will use a database to chart changing taste in American popular music in the Post-War epoch, and Professor Nicholas Temperley will use the ATs to allow students to experiment with data in his mainframe-based hymn index.

Already one student in my spring seminar has decided to continue one of the projects as the basis of her Master's thesis.

We anticipate that familiarity with database programming and availability of these computers will have an increasingly strong impact on teaching and research division. This has been something quite new: most of us did little more than word processing as recently as one year ago. Our future students will take for granted these skills and resources. But right now, Project EXCEL is hastening and facilitating our transition to a new era in musical scolarship.

MUSICAL ACOUSTICS DEMONSTRATION/LESSONS

Presenter: James Beauchamp *University of Illinois*

INTRODUCTION

Musical acoustics is a subject area that can be a challenge to instructors and students alike. Student backgrounds are usually diverse: Some students are knowledgeable about technical or physical matters such as frequency, wave propagation, or signal analysis, while others may be more comfortable with qualitative or psychological aspects, such as sound perception, interpretation, and aesthetics. A musical acoustics course requires the instructor to blend the technical and perceptual issues into a unified structure that conveys the required course material.

To illustrate the difficulties in demonstrating an acoustical concept, consider a single tone played on a violin. A physicist or acoustician might describe the event in terms of frequency in Hertz, frequency spectrum, intensity in Watts per square Meter, duration in Seconds, and so forth. On the other hand, a musician might describe the tone in terms of pitch (G, C#), relative dynamic level (piano, mezzo-forte), note length in beats or fraction of a beat (whole note, eighth note), performance variation such as pizzicato or tremelo, etc. It is essential to reconcile the difference in terminology and emphasis found in the study of musical acoustics by providing objective examples and comparisons of various acoustical phenomena and concepts.

A conventional setup to demonstrate these concepts would include an extensive collection of measurement, recording, playback, and display equipment, along with laboratory space, maintenance capability, and staff to supervise student use. However, this solution is generally impractical due to limited lab space, limited funds, and other reasons. The solution we are working on (with the help of IBM and project EXCEL at the University of Illinois) is to design and program a computer–controlled instrument for demonstration and analysis of acoustic phenomena. Thus, our project is a continuing effort to enhance students' visualization of acoustic concepts through color graphics illustrations and audio input, output, and analysis.

OVERVIEW

The demonstrations/analysis apparatus consists of an IBM PC–AT with several internal hardware expansion boards and peripheral equipment. At this time, the PC–AT is equipped with a 640KB of RAM, an 80287 math co–processor, a 20MB fixed disk and 1.2MB floppy drive, IBM Professional Graphics Controller and Display, a MetraByte Corp. model DASH–16 high–speed analog/digital/analog data acquisition board with direct memory access capability, and Ariel Corp. fast Fourier transform processor board (also with DMA capability), an Octave Plateau OP–4001 Intelligent MIDI Interface board, and two serial/parallel adapter ports for connection to a 6 color IBM 7372 plotter, a Mouse Systems optical mouse, an IBM Proprinter with graphics capability, and a serial communication connection to the LMC MegaMicro Unix–based systems of the University of Illinois Computer Music Project.

The equipment and software is configured for classroom demonstrations, although we eventually plan to make the system available to students outside of class for direct interaction, homework problems, and individual projects. The system is self–contained, and can be transported to and from any classroom using one medium–sized cart.

Five pilot demonstrations are under development, with several others planned for implementation before the Fall of 1986. Three demonstrations (Wave Phenomena, Waveform Synthesis via Fourier Series, and Audio Signal Spectrum Analysis) have been presented so far this semester to the Music 302 – Musical Acoustics course taught by Prof. James Beauchamp. Lessons involving sound output and percussion instrument modes and vibration patterns will be given later. We also expect to develop several demonstrations using the OP–4001 board as a connection to synthesizers and other equipment supporting the Musical Instrument Digital Interface standard, known as MIDI.

The range of input and output devices and the need for a simple and reliable "user interface" has proved challenging for our software development. Other than some low–level device control routines and utility libraries, all of the control software has been designed and implemented from the top on down. The bulk of the software is written in the language 'C' using the Lattice compiler. High–level graphics functions are accomplished using the IBM 'C' language binding for the Graphical Kernal System (GKS), and device drivers of the Virtual Device Interface (VDI). Software requiring high performance, such as real–time digital sampling and analysis or fast animated graphics, is written in assembly language using the IBM Macro Assembler.

With the overview complete, I would like to cover a few salient points regarding the hardware and software.

GRAPHICS -- The Professional Graphics Controller and Display provides 640 x 480 pixel resolution, with up to 256 simultaneous colors chosen from a palette of 4096 different shades. The display controller contains its own 8088 microprocessor and memory, which gives a speed advantage without taking up memory space or bus cycles in the host PC. The resolution and range of colors is not just convenient -- it is really necessary for realistic animation, shading, and waveform plotting (other uses under development include high resolution and music notation). Since hardcopies of graphics are useful for reference or later study, the device independence of the GKS software has been very helpful: a program does not have to be re–written for each of the possible display devices, e.g. plotter, printer, display screen. A simple routing change is made, and the Virtual Device Interface makes the necessary conversions to accommodate the differing characteristics of the output devices.

ANALOG INPUT/OUTPUT -- In addition to the impact of visual demonstrations, audio analysis and synthesis is, of course, of fundamental importance to the study of acoustics. The computer equipped with analog–to–digital and digital–to–analog features provides a way to analyze or sythesize sound in near real–time -- and do so in a reliable, repeatable manner. The MetraByte model DASH–16 board utilizes 12 bit samples, up to 8 differential input channels, and two D/A output channels. The input sampling can utilize direct memory access at a sample rate approaching 50 kHz, while the output D/A (no DMA available) operates at a lower sample frequency. We are in the process of developing software to manage the input data for sustained, continuous sampling over an interval of 10's of seconds, with data going directly to the fixed disk for subsequent manipulation. At present, we are performing steady–state analysis of 1024 sample blocks, obtained in bursts, at 30–40 kHz sample rates. The input may be a line–level signal from any audio source, or from a high impedance microphone through an attached preamplifier.

COMPUTATION AND ANALYSIS -- Many useful demonstration methods and analysis techniques require extensive mathematical calculations. Software running on any PC can perform these calculations accurately, but speed becomes important for effective lessons. One of our goals has been to write demonstration software in such a way that parameter changes can be made "on the fly" during a lesson to show important interrelationships, or in response to student questions. Time spent looking at a blank screen may be tolerable at first, but frequent pauses for recalculation during a lesson can seriously limit the effectiveness of the demonstration. Mathematical operations -- particularly those involving transcendental functions -- are accelerated by the 80287 hardware floating point math co-processor. In several cases we have found the improvement to be such that the calculation delays are insignificant compared to delays in plotting graphics or waiting for user input. The Ariel Corp. PC-FFT fast Fourier Transform processor board performs up to 1024 point forward or inverse FFT's in a few milliseconds. Thus, a signal can be sampled from a microphone, its spectral content or other characteristics calculated, and the result displayed in color within a few seconds.

CONCLUSION

We have spent approximately six months working on this development. As is typical with such projects, we spent the first three months getting acquainted with the software and hardware, chasing down documentation, and planning an implementation schedule. Our experience so far indicates that our goals are attainable, and the demonstrations are useful.

For the near future, we plan to complete programming and documentation for our pilot demonstrations, and experiment with further additions and enhancements. As long term goals, we would like to configure several additional systems for student use outside of class time and for more flexibility in software development and experimentation. This would include development of connections to the existing facilities of the University of Illinois Computer Music Project.

This presentation described our use of an IBM PC-AT system as a demonstration tool and analysis instrument for the study of musical acoustics. We have found the PC-AT's expansion capability useful in our efforts to configure a system that is flexible, portable, reliable, and easy to operate. In short, the system is a useful and effective way to demonstrate acoustical phenomena for an introductory musical acoustics course.

PHILOSOPHICAL SIMULATIONS AND EXPERIMENTS: SCIENTIFIC REASONING PROCESSES

Presenter: Robert L. Causey *University of Texas at Austin*

1. EDUCATIONAL GOALS OF THIS PROJECT

Early Computer Assisted Instruction (CAI) programs largely relied on drill and practice training. Recently, Artificial Intelligence techniques have led to the development of "intelligent tutoring systems", that allow more complex course material and provide more flexible assistance to the student. As the cost of hardware continues to fall, we may expect to see increased use of "intelligent", computer-based tutoring; see [Covey 86], [Anderson 85], and [Johnson 85].

Although CAI and computer-based tutoring systems have significant educational roles, there are other effective, instructional uses of computers. For instance, computers can be used to produce demonstrations that simulate processes or exhibit complex calculations in effective, graphical format. These demonstrations can be run by individual students, or by the instructor during a lecture. I have used such demonstrations in logic and philosophy of science classes for a number of years [Causey 82]. Unfortunately, mere demonstrations do not require active problem solving by the student.

One of the main concerns of philosophy of science is the analysis and codification of various types of scientific reasoning processes. There has been special interest in the general logical structure of scientific explanations, and how the construction of such explanations depends on hypothesis formation and testing. Although almost all undergraduates take some laboratory courses, most of them (including science majors) do not experience the frustrations and joys of creating their own original hypotheses and explanations. Teaching philosophy of science to advanced undergraduates and beginning graduate students is therefore hindered by their lack of experience with novel scientific problem situations and the reasoning processes required to solve them. The principal aim of this project is to use the computer to provide this kind of experience and motivate serious reflection about it.

2. INSTRUCTIONAL USE OF THE SIMULATIONS

For some time computer simulations have been used for research and some teaching purposes. A typical computer simulation models some possible real–world system, for instance, the operation of the human cardiovascular system, [Peterson 84]. Philosophers do not usually perform laboratory experiments, although our work constantly involves thought experiments. It is possible to design a number of useful simulations of abstract relationships and operations. Computer graphics can provide somewhat concrete representations of abstractions, and interactive programs enable interesting computer–simulated experiments. I believe that there are many useful ways to use computers to model abstract problem situations; the programs described in this article are special cases of a broader range of possibilities.

The aim of the current project is to give students experience with scientific problem–solving reasoning. Accordingly, the computer is used to simulate small, generic, scientific research problems. The two programs will be described in more detail in the next section: each program simulates, in terms of colors and geometric shapes, various "worlds" or "systems" which the student investigates. The first program, INDUCT, repeatedly presents nearly–random color patterns in which the randomness is only partially overridden by constraining laws. The student is required to discover, and confirm by experiment, the governing laws. The second program, FAST, presents an abstract system with a few parts. The parts are geometric shapes which can have different colors. The systems (which can be interpreted as factories, biological organisms, etc.) accept an input and produce various outputs, which are also colored geometric shapes. Students working with FAST are presented with the problem of discovering and explaining the functional organization of such systems.

In the case of both programs, the student invents hypotheses to explain the observations and also interactively performs experiments to test his or her hypotheses. The student is required to maintain a paper record of his experiments. Finally, the student writes a short research report which both describes the "experimental research" and "explanatory theory", and which also discusses philosophical questions about the research methodology. Since there is not sufficient time to perform exhaustive tests of the hypotheses, the student cannot be sure that he or she has found the "correct" solution. This is characteristic of scientific research, and it worries the students a little — an intended effect. As a result of all of this, the student acquires experience with:

- Inductive reasoning and hypothesis formation
- Design of experimental tests of hypotheses
- Construction of scientific explanations
- The doubts connected with scientific hypotheses and explanations
- Philosophical analysis of scientific reasoning
- Writing of scientific reports and philosophical discussion

I wish to emphasize two general points. First, for pedagogical reasons, the simulated "worlds" or "systems" are designed with a highly abstract form. This has dual advantages: the students need not, and cannot, rely on knowledge of specific subject matter; the high level of abstraction helps to focus their thinking on the general reasoning processes and the logical analysis of these processes. Second, this instructional use of computers in no way replaces the teacher. The computer is used to extend the student's reasoning experiences in ways that could not easily be achieved by other means.

3. DESIGN AND USE OF THE PROGRAMS

The two programs, INDUCT and FAST, were written in Microsoft TM BASIC and converted into executable code using Microsoft's BASIC Compiler (Version 1.00) and Linker (Version 1.10). The compiled programs are easy for the students to run and satisfy the requirement that the students cannot read the code. The programs run on IBM TM PC's, XT's, and AT's with a color monitor and color graphics adapter board. The graphics display has a black background and uses the standard white, magenta, cyan palette. The programs refer to these colors as "White", "Pink", and "Blue" With no modifications, the compiled programs have run on DOS Versions 2.0 and higher, and on an IBM-compatible computer using MS-DOS TM 3.1. The speed of execution is satisfactory on any of these machines, but is considerably faster with the 80286 processor. The students seem to prefer the faster execution because it speeds up their experimentation.

3.1. THE INDUCT PROGRAM

This program provides experience with inductive reasoning, hypothesis formation, theory testing, and explanation. At start up the student is told that there are several different worlds, identified only by numbers. The student is asked to choose a world by number. World # 1 is reserved for an introductory classroom demonstration that I show the students before they begin their interactive work.

A typical display is indicated by Figure 1. On the monitor screen there is a black background and an outline of 16 squares. Although not shown in the display, each square is identified by an ordered–pair of coordinate numbers; the students refer to a chart showing these coordinates. The coordinates are (x,y), where x is the column number and y is the row number, starting with (1,1) for the upper left square. When the program calls a particular display, the squares are colored white, pink, or blue, starting with square (1,1), then to (2,1), (3,1), (4,1), (1,2), etc., down to the (4,4) square. New displays can be generated indefinitely, simply by hitting the carriage return key. Each new display has a color pattern which is produced almost at random. Yet, depending on the particular world chosen, there will be one or more "laws" that relate the colors of some squares to the colors of some other squares.

At the beginning of each cycle of the infinite loop, a RANDOMIZE COLORS routine generates color codes for each square of the display matrix; a given square is assigned one of the three colors at random. These codes are stored in an internal array in memory; the pattern is not displayed at this point. If the student hits the carriage return, an INVOKE LAWS routine is called. If the antecedent of a law is satisfied by the color codes stored in the internal array, then this law exerts its effect and the color codes of the array are adjusted in accordance with the consequent clause of the law. A trivial example is the law that states: If Pink (1,1), then Blue (4,4). If the array element for square (1,1) has the code for Pink, then this law will set square (4,4) to Blue, regardless of what color (4,4) may have been assigned by the RANDOMIZE COLORS routine. Since the laws of a particular world are just lines of BASIC statements, they are tested for possible firing in the order that they occur in the program. This is a limitation, but it does help to avoid creating an inconsistent world, or laws which can produce endless loops. This flow–of–control limitation also has the conceptual advantage of simulating a simple "causal direction" in the abstract color worlds. In this sequence of events, the net effect is this: the randomization routine initially sets up a random coloration pattern which is not displayed, then the INVOKE LAWS routine may partially override this random pattern by adjusting it according to the laws of the world with which the student has chosen to work. Finally, this lawfully–adjusted color pattern is displayed.

At the bottom of each display the user is prompted for the next action. By pressing the carriage return key (<CR>), the student observes the repeatedly displayed patterns for some time, making notes of what seem to be significant patterns. Eventually, by an inductive reasoning process, or just guesswork, the student forms an hypothesis about a possible law. At this point, he or she can perform experiments to test this hypothesis. When prompted for the next action, he chooses the Set (colors) option, and the SET COLORS routine is called. This routine permits the user to assign particular colors to any desired subset of squares. For example, to make square (2,3) Pink, the student enters "P23". This can be followed by any number of similar entries; after each entry, the list of assignments is displayed and may be modified. This process continues until the student is satisfied with the assignments; then the INVOKE LAWS routine is called, after which the display follows.

The colors randomly assigned to squares are overridden by the student's settings before the INVOKE LAWS routine and the display routine are called. The colors assigned to specific squares by the student will be displayed, except in those cases where a square's color is subsequently changed by the activation of one or more laws. In this way the student can test his or her hypothesized laws by performing a number of experiments. I am a strong believer in the importance of writing practice, so I have the student keep a record of significant experiments, and then write a "research paper" describing significant hypotheses considered, how they were tested, the results that were obtained, and how these results can or cannot explain typical observations. In addition, the paper must relate this "research work" to more general questions about scientific methodology.

3.2. THE FAST PROGRAM

The INDUCT program is designed to simulate "causal laws" of the various color worlds. The student seeks to discover these laws and use them to make "causal explanations" and predictions of the observed color pattern behavior.

Providing this form of deductive explanation from laws plus initial conditions is one way of gaining an understanding of a domain of phenomena.

There are other ways through which we can gain scientific understanding; one of the most common of these is to develop a functional description or functional analysis of a system. If we have a system composed of interacting parts which contribute to the overall behavior of the system, we often want to understand the functional roles of these parts and how these roles contribute to the overall "normal behavior" of the system. Deductive causal explanations and functional analyses are not the only types of scientific explanation, but they are among the most commonly used. Both have been extensively studied by philosophers of science; see, for instance, [Achinstein 83].

FAST ("Functional Analysis System Testing") is designed to provide experience with functional analysis. A typical graphics display of FAST is given in Figure 2. Just as INDUCT offers the student a choice of different color worlds, FAST offers a choice of different systems (one of which is reserved for introductory class demonstrations). Each system has the same overall appearance. There is a constant input to the system, represented by the ellipse on the extreme left. Inside the system (the large rectangle), are four shapes: triangle, square, ellipse, and circle. In a particular display, all, some, and sometimes none, of these will appear. If any one of these figures does appear, it may be White, Pink, or Blue. Finally, the display may or may not exhibit a system output on the extreme right. If an output does appear, it will be an Arrow, a Rod (rod–shaped figure), or a Diamond. Outputs may also be either White, Pink, or Blue.

Although these systems are quite abstract, it is convenient to describe them in familiar language. I suggest to the students that a system be interpreted as a "factory". Inside the factory (the large rectangle) there may be a Boss, a older, a Painter, and a "Messer". The geometric shapes corresponding to these functional positions may vary, both across different systems, and to some extent within a given system. As they are currently designed, each system has constant shapes for its Boss and its Messer. A Boss must be present in order to obtain an output. The color of the Boss determines which shapes will be the Molder and the Painter; this assignment of "duties" is the principal function of the Boss. In this interpretation a system's output object is the "Product" produced by the factory. In general, the color of the Molder determines the shape of the Product, and the color of the Painter determines the color of the Product. One exception to these generalizations is this: in certain special situations the Messer (when present) can interfere with the "normal" operations of the system.

Before a display occurs, a randomization routine pre-selects the colors of the four system parts. In a particular display of FAST, it is also possible that a given part may not appear. In this case, we say that its "color" is "Nocolor". Actually, it is black and so is indistinguishable from the screen background. If the triangle, say, has Nocolor, then the triangle is missing from that display and therefore cannot perform its function. If the triangle happens to be the Boss, then no Product will be produced. By repeatedly pressing the carriage return key, the student can observe various combinations of parts either missing or present in particular colors. At the same time the student can observe what kinds of Products (Rod, Arrow, or Diamond), if any, result, and in what colors. The student's basic problem is to determine, for the chosen system, the rules governing the conditions under which the various shapes play the roles of Boss, Painter, Molder, or Messer, and to determine the functions of these items. The functions are also described in terms of rules: The rules for the Boss say how the color ("disposition") of the Boss determines which shapes it selects for the Painter and Molder. The rules for the Molder state how its color determines the shape of the Product. The rules for the Painter relate its color to the color of the Product. The Messer's rules specify when and how it interferes with the operations of the other rules. Finally, a particular system usually has one or two default rules covering cases not covered by the other functional rules.

These systems can be fairly complicated, even though they have only four internal parts. It would not be convenient to give further details here, but the above description summarizes their general organization. As with INDUCT, the main part of the FAST program runs in an infinite loop. Near the beginning of each cycle of this loop, the randomization routine pre-selects colors, or Nocolor, for each (internal) part of the system. If the student simply presses the carriage return, the rules of that particular system are called, and these rules determine the shape and color of the Product, if any. The system with its colored parts and Product (if any) is then displayed. These randomized display cycles are repeated until the student forms one or more hypotheses about the system's functional organization. At this point he or she can begin serious experimenting, by setting the colors of one or more internal shapes.

The possible settings include Nocolor, which removes that part from the forthcoming display. For example, the student may set Triangle to Blue and Circle to Nocolor. Setting one or more shapes to Nocolor can help to yield evidence regarding which shapes perform essential functions and which do not. The student's settings override the randomized, pre-selected color settings before the rules are called, and then the display occurs. Unlike INDUCT, the student's settings override the randomized settings only in the cases of the particular shapes whose colors he has chosen to set. There is also an option (Keep) for retaining particular settings as long as desired, when cycling through displays with the remaining parts colored randomly.

Working with FAST provides additional experience with inductive reasoning and hypothesis testing beyond that obtained from INDUCT, but FAST is especially helpful in teaching aspects of functional analysis. Although people have experience with functional descriptions in everyday life, this experience is usually not in the context of rigorous experimentation and explanation. FAST provides this rigor while the student is experimenting and when he writes his paper discussing the results. In particular, the student gains experience with

- Hypothesizing the "normal behavior" of a system
- Hypothesizing the functions of the parts of this system, and stating precisely the rules that interrelate these parts as well as their functions with respect to the overall behavior of the system
- Design of experimental tests of functional hypotheses, and the limitations and doubts connected with such hypotheses
- Critical analysis of the logical structure and scientific significance of functional explanations
- Writing of scientific reports and philosophical discussion

4. RESULTS AND DISCUSSION

4.1. PROGRAM DEVELOPMENT AND STUDENT USE

INDUCT was written by the author during the summer and early fall of 1984, and was first used in a graduate philosophy of science class during the fall semester of that year. This class had about a dozen students who were mostly first-year philosophy graduate students, plus a few graduate students from other departments. Almost all of these students were able to complete a very good investigation of one of the color worlds within about two hours of work on the computer. They seemed to find this exercise interesting and enlightening, and they wrote good papers discussing their experiments and related methodological issues.

In the fall of 1985, INDUCT was used by third and fourth year students in an undergraduate philosophy of science class. There were about thirty students in this class, and their reactions were similar to those of the graduate students. Most of the undergraduates also found that two hours of work with the program was adequate, although a few required up to three hours in two working sessions. As one might expect, the undergraduate papers were not, in general, as sophisticated as those of the graduate students, although they were quite good.

Designing FAST required some thought. After the general design was worked out, a student assistant, M. J. Jewell, wrote the first version of FAST during the summer of 1985. Mr. Jewell contributed additional ideas to the overall program design and organization. He undertook this task as part of his senior thesis project for The University of Texas Plan II Liberal Arts Honors Program ([Jewell 85]). After further testing, it appeared that some of the original functional rules in FAST were rather difficult to discover, so I simplified the systems' functional organizations, as well as some other features of the program. FAST was then used in the fall 1985 undergraduate philosophy of science class in November, 1985.

The two programs are neither trivial, nor highly complex. Including comments, INDUCT has 288 lines of source code and FAST has 610 lines. Not counting the time required to develop the general design concepts, there are only two fairly time-consuming aspects of producing programs of this nature. The first is well-known to any applications programmer: a sizable proportion of the program must be devoted to user interface features, including help screens, input error traps, and overall robustness and user-friendliness. These features are especially important in programs that must be immediately usable by students who need to concentrate on their assignments, not on how to run the program. The second special difficulty in writing these programs was developing the "laws" of the color worlds and the "functional rules" of the FAST systems. These laws and rules must be interesting, instructionally effective, yet not too difficult for student use. Considerable time was devoted to inventing and fine-tuning the rules and laws.

Although the students worked with FAST after INDUCT, they found that FAST was more difficult. I think that there are two principal reasons for this: (i) Describing a FAST system requires more rules than a typical INDUCT color world has laws (although the statements of the individual rules and laws are of about equal complexity). (ii) It seems to be conceptually more difficult to provide a systematic functional analysis than it is to write a straightforward deductive explanation from simple "causal laws". Both of these effects were predictable. In fact, the philosophical analysis of functional descriptions, and the kind of explanation based on them, is conceptually troublesome for students, so (ii) was certainly no surprise.

The students seemed to enjoy working with both programs. I am convinced by their efforts, their comments, and the quality of their papers that they learned much from these computer experiments. I was especially pleased by the papers. The writing assignments required considerable philosophical discussion of their experimental methodology as well as more general issues about laws and explanations. For many years I have asked students to write papers about logical and methodological aspects of scientific explanations. My impression is that the papers focused around these two programs exhibited much clearer understanding of some of the basic conceptual issues than those written in previous years without the experience and motivation generated by working with these computer simulations.

4.2. FUTURE OUTLOOK

I plan to use both of these programs in future classes. INDUCT may be augmented with a Keep-settings option, similar to that in FAST, in order to facilitate experimentation. Also, new color worlds and new FAST systems will be designed. I would like to develop an auxiliary program to assist and accelerate the rule-writing process.

I occasionally teach courses on the philosophy of the social sciences and have previously used computer demonstrations of certain types of social interactions in these courses. These demonstration programs are computationally intensive and, until recently, could only be run on large mainframe machines. It may now be feasible to develop similar, interactive simulations that students could use on a fast microcomputer. I am especially interested in the concept of social structure. Analyzing this concept involves considering dispositions of individuals to interact with one another in certain ways. I hope to develop simulations of such phenomena. They will emphasizevery general concepts of interaction, and may only simulate a few simple "social" interactions between semi-autonomous, idealized robots. My aim is to focus on general concepts, rather than particular types of social phenomena.

Although computer simulations are now very familiar, I am convinced that we are just beginning to envision what can be done. We certainly expect to see increasingly accurate simulations of complex, real phenomena. With some imagination and ingenuity, we can also develop simulations that will help us to visualize and reason about conceptually abstract problems. Interactive simulations of this kind could be useful in research by permitting us to experiment with abstract possibilities. They are instructionally useful by giving the student a tangible context in which to exercise general reasoning processes.

ACKNOWLEDGEMENTS

Part of this work, reporting on the INDUCT program, has been previously presented in [Causey 85].

I am indebted to Mr. Michael Jewell for his excellent assistance in the development of FAST. The equipment and software for this project were provided by a grant from The University of Texas Project Quest office, which in turn received a grant from the IBM Corporation. I also wish to thank my students for their hard work with the two programs, and for their useful suggestions regarding future developments.

The contents of this article are not intended as an endorsement of any product. The views expressed are entirely my own and are not necessarily endorsed by any other individual or organization.

REFERENCES

[Achinstein 83] Achinstein, P. The Nature of Explanation. Oxford University Press, New York, N.Y., 1983.

[Anderson 85] Anderson, J. R., and B. J. Reiser. The LISP Tutor. BYTE 10:159–175, 1985.

[Causey 82] Causey, R. L. Microcomputers for Classroom Demonstrations. Issues in Higher Education 6:67–85, 1982.

[Causey 85] Causey, R. L. Microcomputer Simulations in the Teaching of Logic and Philosophy of Science. 1985. Invited paper presented at the conference on Logic, Logic Machines, and Public Education, sponsored by The Institute for Logic and Cognitive Studies, University of Houston–Clear Lake, June 13–15, 1985. Forthcoming in the conference proceedings.

[Covey 86] Covey, P. K. Computer–Assisted Instruction in Philosophy. In S. Olsen (editor), Computer–Assisted Instruction in the Humanities. Modern Language Association, forthcoming 1986.

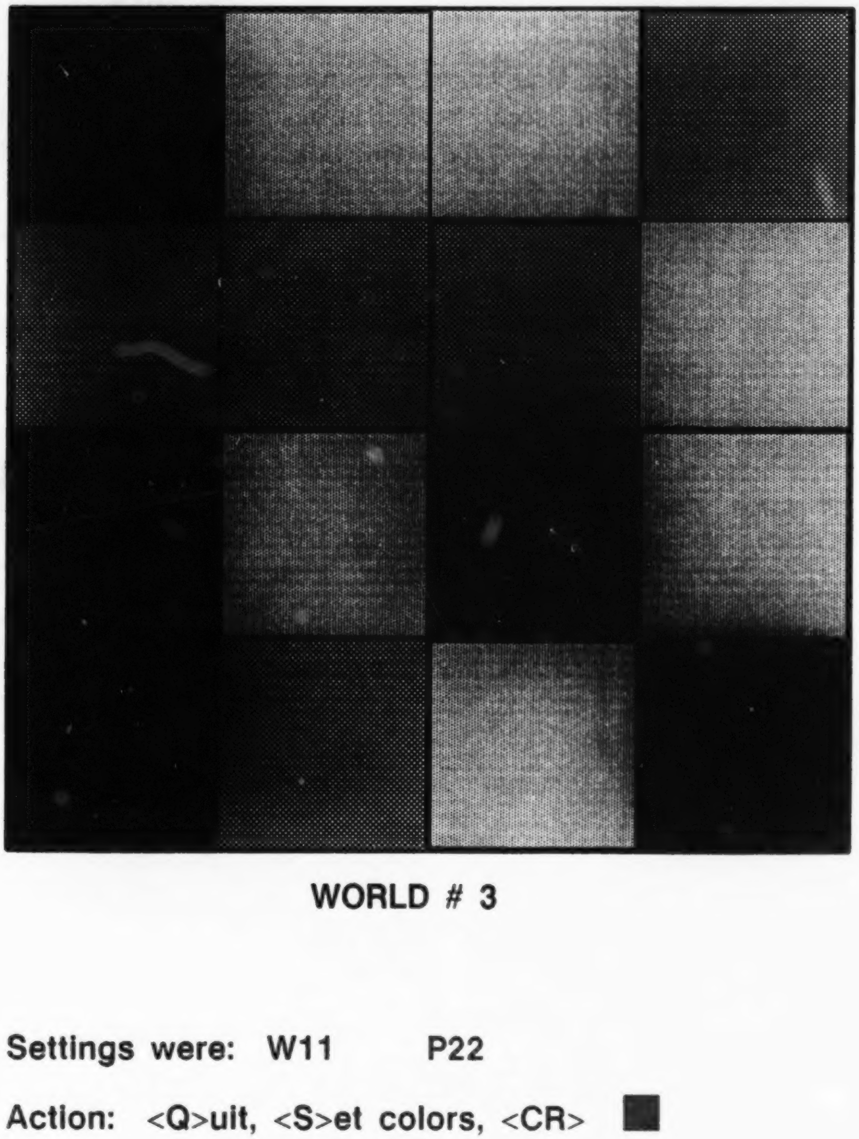
[Jewell 85] Jewell, M. J. Computerized Functional Analysis Teaching System. August, 1985. Senior Thesis for the University of Texas at Austin Plan II Liberal Arts Honors Program.

[Johnson 85] Johnson, W. L., and E. Soloway. Proust. BYTE 10:179–190, 1985.

[Peterson 84] Peterson, N. Designing a Simulated Laboratory. BYTE 9:287–296, 1984.

Figure 1:
A typical display of INDUCT

The different shadings represent the colors pink, blue, and white. This figure reverses screen occurrences of black and white.



WORLD # 3

Settings were: W11 P22

Action: <Q>uit, <S>et colors, <CR>

REMOTE ELECTRONIC MUSIC SYNTHESIS WORKSTATIONS

Presenter: Russell Pinkston *University of Texas*

copyright © Russell F. Pinkston, 1986

INTRODUCTION

An electronic and computer music facility of some kind is now a necessity for any self-respecting graduate (or undergraduate) degree program in music composition. Commercially available digital synthesizers, which are becoming more and more powerful, but less and less expensive on an almost daily basis, are rapidly transforming the traditional electronic music center. These new machines, while they offer many new opportunities for serious composers, are generally designed with the commercial musician in mind; they tend to be remarkably "user friendly" and come equipped with libraries of "canned" electronic sounds, but all too often they severely limit the musical/timbral options available to the user. The training of composers, especially those who are planning academic careers, must go far beyond teaching them how to use these facile machines to play pre-fabricated sounds. It is imperative that we also teach them enough about sound, itself, the various synthesis and processing methods available to create and manipulate it, and how synthesizers implement those methods, that they can master these new instruments and use them intelligently to realize their compositions. Teaching composers how to synthesize their sounds in software, using general purpose computers running music programs the students can design themselves, is an ideal way to ensure that they will become competent "electronic musicians" and that, as composers, they need never be trapped by what someone else's idea of "music" is. As composer Paul Lansky has said, "a general-purpose computer is literally the most powerful musical instrument ever designed....

> With practice, as in learning to play any other musical instrument, a composer can have a real 'instrument of imagination' at his fingertips."

Synthesizing and processing music in software, however, requires a large, fast computer, and specialized hardware for digital-to-analog and analog-to-digital conversion of high fidelity sound. This equipment is expensive enough that, until recently, only large, research-oriented (and research grant funded) computer music centers could afford it. The University of Texas at Austin recently opened a new Computer Music Studio, which uses IBM PC/ATs as Electronic Music Synthesis Workstations. The PCs are linked to an IBM 3081 mainframe, owned and operated by our Computation Center. The purpose of this paper is to describe the new facility in some detail, to explain the rationale behind the specific configuration we chose, and to discuss its purposes, present capabilities and future potential.

BACKGROUND

The Department of Music at The University of Texas at Austin is the largest in the country, with 80 faculty members and over 800 students. Our Electronic Music Center is one of the most extensive, as well. We currently operate five separate studios and typically have forty or more students and faculty using them each semester. In contrast to such research-oriented centers as those at Stanford, UCSD, MIT, etc., we think of ourselves as being, first and foremost, a center for the composition, performance, and teaching of music. In designing our new Computer Music Studio, therefore, our primary goal was to provide a creative facility intended for use by persons trained in music, not necessarily in physics, engineering, or computer programming. This implied that it should be as easy to use as possible, but also that it should be capable of satisfying the often demanding requirements of sophisticated, imaginative, classically trained musicians and composers. The ability to hear the results of one's experiments with a minimum of delay and in an environment conducive to creative thought was felt to be of paramount importance. It was also considered to be desirable to maintain compatibility with other computer music centers whenever possible, and to choose musical flexibility over user-friendliness if direct conflicts arose. Of course, one of the most frequent and demanding requirements of all composers is for more time in the studio! Consequently, our new computer music facility also needed to be able to handle as many users as possible, given the size and scope of our particular program.

CONFIGURATION

The heart of our system is an IBM 3081-D16 mainframe computer, running the VM/SP operating system, that is housed in the U.T. Graduate School of Business, approximately 1/2 mile from the Music Department. This is an extremely fast computer (10 - 12 million instructions per second), that is ideal for running the highly cpu-intensive music synthesis and signal processing programs. It also has over 35 Gigabytes of disk storage, 16 megabytes of real memory, practically unlimited virtual memory, and it can comfortably handle 400 to 450 users simultaneously. The greatest advantage it offers to the Department of Music is neither its power, nor its storage capacity, however, but the fact that we did not have to buy it, and do not have to maintain it! To provide convenient access to the 3081, we have three IBM PC/AT Model 99s housed in the Electronic Music Center and local attached to the 3081 via co-axial cable. Each of our ATs has an IBM 3278/9 Emulation Adapter card and Control Program, which enables it to function as an IBM 3278 terminal and to perform file transfers to and from the 3081. The ATs are also equipped with IBM Data Acquisition and Control Adapters, each of which has two 12-bit digital to analog converters onboard. Custom software was written to allow users to perform stereo digital to analog conversions from memory. (It turned out not to be possible to perform audio rate conversions other than from memory, due to the fact that the AT does not use DMA for transfers involving its hard disks.) Users synthesize and process their sounds on the 3081, using the AT for terminal emulation, and then transfer them to the AT as a large binary file, which is stored on one of the two 20 megabyte hard disks. The user then "escapes" back to DOS and performs a digital to analog conversion on the sound file. The Studio is equipped with anti-aliasing filters, DBX noise reduction, and professional quality tape decks, so that it is a simple matter to record any sounds the user wishes to keep.

PERFORMANCE

The performance specifications that are likely to be of interest to computer musicians are the following:

FILE TRANSFERS

File transfer speed is highly dependent on the 3081 system load. We have not yet run official benchmarks, but during peak hours (weekday afternoons), binary file transfers seem to average about 1K bytes per second on the AT. During off-peak hours, they seem about 20 to 50% faster. By packing the data (two 12-bit samples per 3 bytes), throughput rates of between 650 and 1000 samples per second are consistently achievable. This means a wait of between 15 and 25 seconds for each second of monaural sound at a 15KHz sampling rate. This is certainly far from ideal, but when one considers the fact that the complexity of the sounds is completely unrelated to the transfer rate -- 10 seconds of Beethoven's Missa Solemnis takes the same length of time to transfer as 10 seconds of a pure sine wave -- they are satisfactory for testing purposes. It is important to bear in mind that the length of time it takes to compute the sounds is vastly reduced by using a processor like the 3081. Attempting to synthesize directly on a PC, even one as fast as the AT, might mean hours of waiting for just a few minutes of sound. At our studio, users typically run short tests of the sounds they are working with on the 3081, wait a few minutes while they are transferred to the AT, convert them, and then edit their programs and repeat the process. When a user has completed the testing process and is ready to hear an extended section of music, he can either write the output to digital tape and walk across campus to retrieve it, or start the file transfer and perform some other studio task while it completes.

D–TO–A CONVERSIONS

Using the IBM Data Acquisition and Control Adapters, an IBM PC/AT with 640K of Base Memory, and our own Disk I/O and D–to–A Conversion software, we can convert up to 274,000 samples at a time at maximum sampling rates of 30KHz mono and 24KHz stereo. Our Disk I/O routines are fast enough to fill the sample buffer (558K bytes) from files on the hard disks in about 3 seconds, so that, while it is not possible to convert extended sound files in a continuous run, it is possible to "step" through them quite rapidly. We are currently developing software to allow D–to–A conversions from the extended memory of the ATs, which would allow us to increase the number of samples per run considerably, but it will probably never be possible to convert directly from disk with this particular hardware configuration, for reasons mentioned above. To convert extended pieces of continuous music, the user writes a digital tape on the 3081, and converts it on a different system at this time.

POSSIBLE ENHANCEMENTS

The configuration described above is more than adequate to handle the task for which it was intended, namely, to serve as an editing/sound testing facility for computer musicians using an IBM mainframe to run their music programs, but intending to use some other system to play and record their finished pieces. It is not adequate, however, to serve as a "complete" computer music system. The principal weaknesses are the limited dynamic range (a 12–bit converter yields only about 72dB) and the fact it is not possible to convert more than a limited number of samples at a time. With some additional hardware and software, however, the system could easily be upgraded. Some suggestions follow:

Purchase a set of 16–bit A–to D and D–to–A converters with DMA interface for IBM PC/AT. (At least one company offers such a package, with software and anti–aliasing filters, for about $5,000.) This single enhancement would permit continuous A–to–D and D–to–A conversions from hard disk with 96dB dynamic range and acceptable sampling rates in stereo.

Purchase a 1600/6250 BPI 1/2–inch streaming digital tape drive with DMA interface for IBM PC/AT. (There are several such drives on the market.) This would enable users to write large files to tape in a format readable by both mainframe and PC; if the drive were sufficiently fast, conversions could be performed directly to and from the tape. This would make the length of continuous conversions limited only by the capacity of a single reel of tape, which is quite large. It would also provide a convenient means of transferring very large files between PC and mainframe, not to mention sound file archiving.

Purchase light pen and/or mouse for interactive graphics, etc.

Purchase MIDI interface card and software to allow real–time performance input and control from MIDI devices.

SOFTWARE FOR PRINTING SOUTH ASIAN SCRIPTS

Presenter: Donald A. Becker *University of Wisconsin–Madison*

While teachers of French, German, Spanish, and Russian can draw upon a wide range of excellent textbooks, grammars, and dictionaries dealing with their subject, their colleagues who teach South Asian languages are far less fortunate. Instructors of Hindi, Urdu, Telugu and other languages of the Indian Subcontinent often find themselves quite hard–pressed to find adequate readings, grammatical exercises, and bilingual glossaries for their students. To compensate for the shortage of such printed resources, scholars specializing in these languages have been forced over the years to devote an inordinate amount of their time and energy to the production of supplementary teaching materials. Although in terms of content these materials have often been excellent, their outward appearance has left much to be desired, since the lack of typesetting facilities for Indic scripts in this country has forced their authors to write them out in longhand or type them on antiquated typewriters capable of producing only a very rough approximation of a good typeset text. The production problems have been compounded by the need to intermix two writing systems on the same page: that of that target language and the Roman script of the English glosses and grammatical notes. Since daisywheels and Selectric–style typing elements do not exist for the languages in question, writers of instructional materials have had to move each page of their manuscripts back and forth many times between a standard American typewriter and an Indic–script machine. The typographical unsightliness of works produced in this manner has generally prevented their publication by photo offset. They have, instead, been distributed (in Xeroxed or Dittoed form) only to their author's own students. Their potential value has accordingly not been realized.

Our project is attempting to show that an IBM microcomputer can help alleviate the chronic shortage of teaching materials for the languages of South Asia by providing teachers of those languages with a convenient and inexpensive means of publishing some of the useful pedagogical manuscripts which they have developed with much care but little hope of ever seeing in print. The project involves the construction of printer driver programs in compiled BASIC that enable their users to compose South Asian texts in Roman transliteration with the aid of standard word processing software for the IBM PC, save these texts in

ASCII format files, and then print out, on a high-resolution dot matrix printer, a final camera-ready copy on which all non-English words are transliterated into one of three target scripts. These three scripts are (1) Devanagari, in which such languages as Hindi, Marathi, Nepali, and Sanskrit are written; (2) Telugu, which is used for the Telugu language as well as for Sanskrit by speakers of Telugu and other Dravidian languages; and (3) the specially modified variety of Arabic script which has been developed for Urdu. Specimens of these three writing systems produced by my printer driver programs can be seen in figure 1. Next to each print sample is a verbatim listing of the Romanized input from which it was generated by the computer controlling the printer.

Although the choice of the three target scripts was determined mainly by the language curriculum of the Department of South Asian Studies at the University of Wisconsin, these scripts are actually quite well-suited to serve as the basis of a pilot project of the kind we are undertaking. From a typological standpoint, it would be difficult to find three writing systems of the Indian Subcontinent that are more diverse in appearance or structure. The Devanagari and Telugu scripts are written from left to right, whereas the Arabic script of Urdu runs from right to left. Both of the former scripts are syllabaries (i.e. the basic unit represented is the syllable), while the latter is a kind of shorthand, in which short vowels are normally not represented. Devanagari employs the horizontal "matra" line common to several writing systems of Northern India, the other two scripts do not. All three scripts exhibit more than their share of complexities. Extending our approach to other writing systems of the region is therefore unlikely to pose any problems not yet encountered in at least one of our three.

The use of the Roman alphabet to represent the target languages on the screen of the computer's video display was dictated by various considerations. Foremost among these, however, was my realization that by dispensing with any attempt to display the scripts on the screen, I would be able to focus all of my attention on the task of getting the most out of the dot matrix printer. The amount of programming time saved in this manner has been enormous, for had I decided to try to bring the target scripts to the screen, I would have had to design fonts for the video display as well as for the printer. Worse yet, I would have been forced to construct a set of text editor programs capable of manipulating the complex target scripts on the screen. Given my limited programming experience, such programs would have necessarily been rather primitive. On the other hand, by contenting myself with a Romanized text on the screen, I have been able to take full advantage of most of the sophisticated text–editing features offered by some of the best word processing packages on the market with no investment of programming time whatsoever.

My printer driver programs share a common structure. Although the three target scripts differ markedly in appearance and organization, they are similar in one crucial respect: Each of them represents a particular sound differently in different contexts. Whereas traditional typewriters had to assign the variant shapes of a particular grapheme to different locations on the keyboard, my printer drivers are designed to select automatically the correct variant (or "allograph") of each grapheme for each environment in which it occurs. It is precisely the ability of the computer to determine which of several allographs is appropriate in a given context that makes the use of a Romanization as an input vehicle a viable programming option. If not only every grapheme, but every allograph of every grapheme had to be represented by a distinct ASCII character, the resulting Romanization would be much too cumbersome to learn and work with. A Romanization in which only distinct graphemes are represented, on the other hand, is quite easy to learn, for it closely approximates a phonemic transcription of the language in question. The selection of allographs is accomplished as follows: The program in question scans each line of the Romanized text in the disk file from beginning to end (or in the case of Urdu, which is printed backwards, from end to beginning). As each grapheme of the line is encountered, the program branches to a subroutine containing context–sensitive rules that determine which of a number of possible representations of the sound in question is appropriate in the current environment. To facilitate this scanning process, each grapheme in the line is first assigned to an element of a one–dimensional array. The subscripts of the various elements in that array indicate the order in which the graphemes they represent are to be printed. The assignment of a particular grapheme to a particular array element must therefore take into account certain idiosyncrasies of the target script. In the Telugu syllabary, for example, the most common symbol for a post–consonantal "r" is written before, i.e. to the left, of the consonant it follows. If a particular Telugu consonant is assigned to the array element A$(n), then the "r" which follows it in pronunciation and hence in the Romanization must be assigned to the element A$(n–1). In the Urdu (i.e. Arabic) script, double consonants are written only once with an optional diacritic to indicate the fact that they are geminates. Both Urdu consonants must accordingly be assigned to the same array element. In the Devanagari orthography of Hindi, contiguous consonants are frequently merged into a single complex shape. Here again, proper sequencing demands that several characters in the Romanization be assigned to a single array element.

The dot matrix printer which I am currently using is a Toshiba P1340. Although the resolution of its output is inferior to that of a laser printer or of IBM's new QuietWriter II with which I will be working in the coming months, I have been able to enhance its print quality considerably by making the printhead double–print each line. Since portions of certain large characters are printed on successive lines, I have disabled bidirectional printing. This insures the best possible alignment of the components of these characters. The quality of the printed output depends, however, above all on the relatively high density of the dot pattern. The 24–pin printhead of the P1340 prints a very respectable 180 dots per inch both vertically and horizontally. From a programmer's standpoint, the high number of dots composing each character means that a great many instructions must be transmitted from the host computer to the printer for each symbol of the target script. Just how many instructions we are dealing with in the case of the Toshiba P1340 is illustrated by figure 2. Figure 2 shows how one derives the 124 codes needed to address the pins which must be fired in order to print the Telugu syllable "ha", the last character of the Telugu syllabary. For this purpose, the 24 pins, which print each vertical slice of that character, are divided into four groups of six pins each. One can think of each group of 6 pins as a six–digit binary number in which a 1 indicates that the pin in question is to be fired and a 0 indicates that it is not. The top pin in each group represents the highest power of 2, the bottom pin the lowest. The resulting number is transmitted to the printer in decimal or hexadecimal form as the argument of the BASIC CHR$(X) function. This notation is obviously rather awkward to use. Moreover, it takes up a large amount of valuable memory space. Programming as I am in IBM Disk BASIC, I have to be ever mindful of the 64K limit on the size of my source code, beyond which I have to begin chaining programs together. I have therefore found it convenient to define at the beginning of my printer driver programs a series of string constants whose names consist of one or more of the first six letters of the alphabet. The presence of a particular letter in the name of a string constant indicates that the corresponding pin is to be fired when that string name appears after BASIC's LPRINT command. For example, the letter "A" in a string constant's name indicates that the lowest pin of a group of six is to be fired. Similarly, the letter "F" activates the top pin. The combination "BCD" signals that the second, third, and fourth pins from the bottom are to strike the page simultaneously.

Although motivated primarily by the desire to save memory space, the use of this abbreviative notation makes graphics programming considerably more intuitive and my source code far easier to read.

In addition to making it possible for South Asian scholars to engage in desktop publishing of textbooks and readers in three of the most important writing systems of the region, our project is seeking to exploit the power of the microcomputer to help offset the critical shortage of reliable bilingual dictionaries suitable for use by English-speaking students of South Asian languages. This phase of the project involves the preparation of alphabetizing utilities for the target scripts. These utilities, when used with other short programs that merge pairs of alphabetized lists, enable a language teacher to construct quickly and easily a comprehensive glossary for a set of lessons that he or she has devised. Thus far, a complete set of three utilities has been developed for Telugu. The first program derives from a text file of Romanized Telugu a new file in which all of the non-English words occurring in the text are listed in the alphabetical order used in most Telugu dictionaries. The derived file contains one word per line. Duplicate occurrences of the same word are automatically eliminated. The instructor can then use standard word processing software to insert appropriate English glosses after each entry. The second program merges the vocabulary lists derived from two texts into a single alphabetized list. The third program merges pairs of alphabetized lists to which English glosses have already been added. The bilingual wordlists prepared in this manner can be printed in the proper scripts with my Telugu printer driver. All Telugu and English words appear on the printout in the appropriate script.

When I first decided to write alphabetizing programs for the three scripts targeted by our project, the task appeared to me to be quite formidable. After all, the "greater than" and "less than" comparisons that form the basis of sorting programs for lists of English words can only work because the ASCII codes associated with various letters reflect the ordering principles used in English dictionaries. The order of letters in each of the target scripts differs, however, quite radically from the alphabetical order familiar to English speakers. I was beginning to work out elaborate sorting strategies when it dawned upon me that a simple programming trick would enable me to use any alphabetizing program available for English with Romanized Hindi, Urdu, or Telugu. That trick consisted in replacing the Roman letters of all the words to be sorted with letters whose ASCII values reflected the relative positions of the original letters in the alphabetical order of the Devanagari, Telugu, or Urdu writing systems. A hypothetical example will serve to illustrate this procedure. Suppose that for alphabetizing purposes, we wished to pair off the letters in the Roman alphabet and then reverse the order of the members of each pair. Words beginning with "B" would then precede those beginning with "A", words beginning with "D" would precede those beginning with "C", etc. Without writing new alphabetizing programs or reassigning the ASCII codes assigned to each letter, we could perform the new kind of alphabetization by replacing each original "A" with a "B", each original "B" with an "A", each original "C" with a "D", each original "D" with a "C", and so on. We could then alphabetize the encoded words with any of our favorite old-style sorting programs. Finally, we would decode the alphabetized list generated by that sorting program by reversing the substitutions made prior to sorting. While this sounds rather complicated, the BASIC code listed in figure 3 shows how simply the encoding and decoding required by this example can be accomplished. Alphabetizing Romanized Hindi, Urdu, and Telugu words in accordance with the ordering principles of their respective writing systems is, in essence, no more difficult than the situation outlined here.

In conclusion, I would like to mention two things that our project has taught me about writing academic software. The first lesson I have learned is that users will apply programs to tasks that were never anticipated by the programmer. A good example of this has been provided by the Telugu instructors at the University of Wisconsin. The alphabetization utilities described above were designed to help language teachers prepare glossaries from the text files containing their pedagogical manuscripts. Our Telugu instructors, however, have found a way of letting their more advanced students share in the work of producing teaching materials. They have trained the students to type Telugu texts in the Romanization assumed by my Telugu printer driver program and then asked each student to convert a few pages of a Telugu folk epic published in India into machine-readable form. The resulting vocabulary files have been alphabetized and merged together using my glossary-building software. The instructors are now filling in appropriate English glosses after each entry in this comprehensive word list. When they have finished this task, they will have created a tool that will make a work of literature accessible to students of Telugu who previously could not have possibly read it in the original. The students do not seem to find the job of transliterating texts to be an imposition. On the contrary, the fact that they are contributing to the development of materials that will be of real value both to them and to future Telugu classes adds a participatory dimension to their Telugu course that they truly seem to enjoy.

The second lesson I have learned is that no matter how much time the programmer devotes to perfecting a piece of software, its users will always be able to suggest numerous ways to improve it. Some of the improvements suggested by users of my programs have been relatively easy to incorporate. I have customized the Romanization used with all three printer drivers to accommodate preferences of their main users. Embellishments such as tab stops and a draft printing mode have been built into the Telugu printer driver in response to wishes of the Telugu staff. I have not, however, been able to satisfy the wish heard most often from my programs' users. I am referring here to the desire of nearly every user to see the target scripts on the video display as well as on the printed page. Earlier in this paper, I explained that my chief reason for choosing a Romanization as an input vehicle was that it greatly simplified my job as a programmer. This explanation is, however, hardly calculated to impress the typical program user for whom the principle "what you see is what you get" has come to be regarded as a litmus test of user friendliness. The most effective way I have found to reply to those who yearn to see exotic symbols dancing across the screen, is to show them the alternative to my printer driver approach embodied in the South Asian fonts for the Apple Macintosh developed by several South Asianists, including myself. I point out that the visual feedback offered by these fonts is achieved only by sacrificing other, highly valued features of my Romanization–based approach. One such feature is the standard QWERTY keyboard. Every Indic font for the Macintosh with which I am familiar imitates traditional typewriters in representing all the allographs of each grapheme on the keyboard. The font's users are thus obliged to master a new and highly complex keyboard layout for the script in question. In the process, even skilled typists are reduced to the hunt–and–peck method of data entry. To make matters worse, the structures of the Indic scripts are sufficiently varied that one cannot employ the same keyboard layout for all of them. The ease with which the Macintosh can replace one Roman font with another does not, therefore, extend to the substitution of one Indic font for another. Type a Sanskrit passage in Devanagari script on a Macintosh, highlight it, and switch to a Telugu font: The result will be gibberish! By contrast, my Romanization–based approach requires no new typing skills. Moreover, the use of a neutral Romanization allows one to adapt the script in which a particular text is printed quickly and easily to the needs of a specific audience. One only has to select the appropriate printer driver. It is true that arbitrary spellings in one or the other script occasionally make it necessary to edit a Romanized text file slightly before its contents are printed in a second script. However, the amount of effort required by such editing is minuscule compared to the drudgery of retyping the entire text, using a totally different keyboard arrangement for the second script. I thus feel justified in claiming that my Romanization–based approach is more user–friendly than the font–based approach implemented on the Macintosh. Someday someone will no doubt devise a smart word processing package for the Macintosh that will enable one to type South Asian scripts in graphemes rather than allographs. Until then, the advantages of using a Romanization for input and editing purposes clearly outweigh any advantages associated with seeing the target script on the screen.

Software for Printing South Asian Scripts

Donald A. Becker, Departments of German and Linguistics, University of Wisconsin-Madison

Figure 1: Romanized Input Compared with Printed Output in Hindi, Urdu, and Telugu:

jAnwaro~ me~ gadHA sabse z+yAdA bud+dHihIn samjHA jAtA hE. ham jab kisI AdmI ko pahle dar+je kA bewakUf kahnA cAhte hE~, to use gadHA kahte hE~. gadHA sacmuc bewakUf hE, yA uske sIdHepan, uskI nirApad sahi$+NutA ne use yah padwI de dI hE, iskA niS+cay nahI~ kiyA jA saktA. gAye~ sIMg mArtI hE~, b+yAyI huI gAy to anAyAs hI siMhnI kA rUp dHAraN kar letI hE. kut+tA bHI bahut GarIb jAnwar hE, lekin kabHI-kabHI use bHI k+rodH A hI jAtA hE; kin+tu gadHe ko kabHI k+rodH karte nahI~ sunA, na dekHA.

जानवरों में गधा सबसे ज़्यादा बुद्धिहीन समझा जाता है। हम जब किसी आदमी को पहले दर्जे का बेवक़ूफ़ कहना चाहते हैं, तो उसे गधा कहते हैं। गधा सचमुच बेवक़ूफ़ है, या उसके सीधेपन, उसकी निरापद सहिष्णुता ने उसे यह पदवी दे दी है, इसका निश्चय नहीं किया जा सकता। गायें सींग मारती हैं, ब्यायी हुई गाय तो अनायास ही सिंहनी का रूप धारण कर लेती है। कुत्ता भी बहुत ग़रीब जानवर है, लेकिन कभी-कभी उसे भी क्रोध आ ही जाता है; किन्तु गधे को कभी क्रोध करते नहीं सुना, न देखा।

جانوروں میں گدھا سب سے بیوقوف سمجھا جاتا ہے۔ ہم جب کسی شخص کو پہلے درجے کا احمق کہنا چاہتے ہیں تو اسے گدھا کہتے ہیں۔ گدھا واقعی بےوقوف ہے یا اس کی سادہلوحی اور انتہا درجہ کی قوّت برداشت نے اسے یہ خطاب دلوایا ہے۔ اس کا تصفیہ نہیں ہو سکتا۔ گائے شریف جانور ہے مگر سینگ مارتی ہے۔ کتا غریب جانور ہے لیکن کبھی کبھی اسے بھی غصّہ آ جاتا ہے۔ مگر گدھے کو کبھی غصّہ نہیں آتا۔

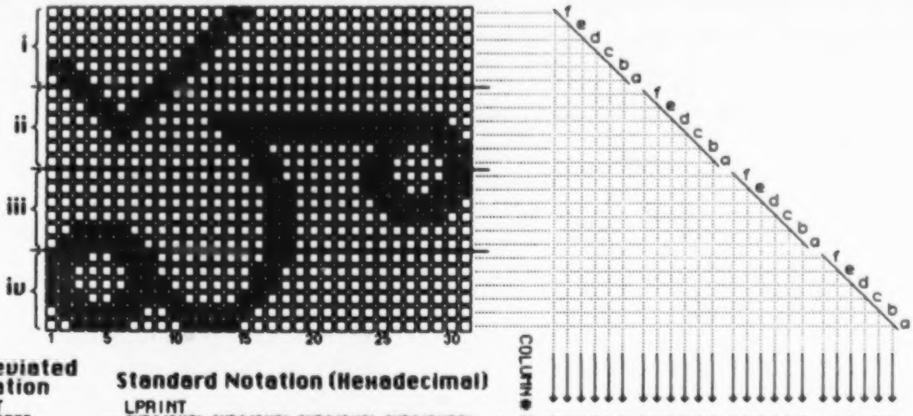
jAnwaro~ me~ gadhA sab se bewaqUf samjhA jAtA hE. ham jab kisI s^axS ko pahle darje kA aHmaq kahnA cAhte hE~ to use gadhA kahte hE~. gadhA wAqi`I be#waqUf hE yA us kI sAdah#lOHI Or intihA darjah kI quwwat-e bardAs^t ne use yih xit_Ab dilwAyA hE. is kA taSfiyah nahI~ ho saktA. gA'e s^arIf jAnwar hE magar sIng mArtI hE. kuttA GarIb jAnwar hE lekin kabhI kabhI use bhI GuSSah A jAtA hE. magar gadhe ko kabhI GuSSah nahI~ AtA.

bhaaratadeeSamaMdugala goppa nagaramulaloo haidaraabaadu okaTi. ii nagaramu muusii anu nadi yoDDuna nirmiMpabaDiyunnadi. muusiinadi kr+i$Naanadi yokka upanadiyee. ii nagaramunu guurci aneeka kathalunnavi. ibrahiMkutub Saahi anu navaabu goolakoMDanu paaliMcucuMDenu. goolakoMDa prastutapu haidaraabaadunaku ceeruvagaa nunnadi.

భారతదేశమందుగల గొప్ప నగరములలో హైదరాబాదు ఒకటి. ఈ నగరము మూసీ అను నది యొడ్డున నిర్మింపబడియున్నది. మూసీనది కృష్ణానది యొక్క ఉపనదియే. ఈ నగరమును గూర్చి అనేక కథలున్నవి. ఇబ్రహింకుతుబ్ శాహి అను నవాబు గోలకొండను పాలించుచుండెను. గోలకొండ ప్రస్తుతపు హైదరాబాదునకు చేరువగా నున్నది.

Figure 1.

Figure 2: Digitizing the Telugu Character హ [ha]:



```
Abbreviated
 Notation            Standard Notation (Hexadecimal)
LPRINT               LPRINT
B;0;0;CDEF;        = CHR$(&H2);CHR$(&H0);CHR$(&H0);CHR$(&H3C);   ←1) 000010 000000 000000 111100
AB;0;A;BCDEF;      = CHR$(&H3);CHR$(&H0);CHR$(&H1);CHR$(&H3E);   ←2) 000011 000000 000001 111110
A;F;AB;ABCF;       = CHR$(&H1);CHR$(&H20);CHR$(&H3);CHR$(&H27);  ←3) 000001 100000 000011 100111
0;EF;AB;AB;        = CHR$(&H0);CHR$(&H30);CHR$(&H3);CHR$(&H3);   ←4) 000000 110000 000011 000011
0;DE;AB;AB;        = CHR$(&H0);CHR$(&H18);CHR$(&H3);CHR$(&H3);   ←5) 000000 011000 000011 000011
0;CD;AB;ABCF;      = CHR$(&H0);CHR$(&HC);CHR$(&H3);CHR$(&H27);   ←6) 000000 001100 000011 100111
0;DE;A;BCDEF;      = CHR$(&H0);CHR$(&H18);CHR$(&H1);CHR$(&H3E);  ←7) 000000 011000 000001 111110
0;EF;0;CDEF;       = CHR$(&H0);CHR$(&H30);CHR$(&H0);CHR$(&H3C);  ←8) 000000 110000 000000 111100
A;F;0;CD;          = CHR$(&H1);CHR$(&H20);CHR$(&H0);CHR$(&HC);   ←9) 000001 100000 000000 001100
AB;0;0;BC;         = CHR$(&H3);CHR$(&H0);CHR$(&H0);CHR$(&H6);    ←10) 000011 000000 000000 000110
BC;0;0;AB;         = CHR$(&H6);CHR$(&H0);CHR$(&H0);CHR$(&H3);    ←11) 000110 000000 000000 000011
CD;0;0;AB;         = CHR$(&HC);CHR$(&H0);CHR$(&H0);CHR$(&H3);    ←12) 001100 000000 000000 000011
DE;D;0;AB;         = CHR$(&H18);CHR$(&H8);CHR$(&H0);CHR$(&H3);   ←13) 011000 001000 000000 000011
EF;CD;0;AB;        = CHR$(&H30);CHR$(&HC);CHR$(&H0);CHR$(&H3);   ←14) 110000 001100 000000 000011
F;BCD;0;BC;        = CHR$(&H20);CHR$(&HE);CHR$(&H0);CHR$(&H6);   ←15) 100000 001110 000000 000110
0;ABCD;F;CDE;      = CHR$(&H0);CHR$(&HF);CHR$(&H20);CHR$(&H1C);  ←16) 000000 001111 100000 011100
0;ACD;ABCDEF;FED;  = CHR$(&H0);CHR$(&HD);CHR$(&H3F);CHR$(&H38);  ←17) 000000 001101 111111 111000
0;CD;ABCDE;F;      = CHR$(&H0);CHR$(&HC);CHR$(&H1F);CHR$(&H20);  ←18) 000000 001100 011111 100000
0;CD;0;0;          = CHR$(&H0);CHR$(&HC);CHR$(&H0);CHR$(&H0);    ←19) 000000 001100 000000 000000
0;CD;0;0;          = CHR$(&H0);CHR$(&HC);CHR$(&H0);CHR$(&H0);    ←20) 000000 001100 000000 000000
0;CD;0;0;          = CHR$(&H0);CHR$(&HC);CHR$(&H0);CHR$(&H0);    ←21) 000000 001100 000000 000000
0;CD;0;0;          = CHR$(&H0);CHR$(&HC);CHR$(&H0);CHR$(&H0);    ←22) 000000 001100 000000 000000
0;CD;0;0;          = CHR$(&H0);CHR$(&HC);CHR$(&H0);CHR$(&H0);    ←23) 000000 001100 000000 000000
0;ACD;EF;0;        = CHR$(&H0);CHR$(&HD);CHR$(&H30);CHR$(&H0);   ←24) 000000 001101 110000 000000
0;ABCD;FED;0;      = CHR$(&H0);CHR$(&HF);CHR$(&H38);CHR$(&H0);   ←25) 000000 001111 111000 000000
0;BCD;CDE;0;       = CHR$(&H0);CHR$(&HE);CHR$(&H1C);CHR$(&H0);   ←26) 000000 001110 011100 000000
0;CD;CD;0;         = CHR$(&H0);CHR$(&HC);CHR$(&HC);CHR$(&H0);    ←27) 000000 001100 001100 000000
0;CD;CD;0;         = CHR$(&H0);CHR$(&HC);CHR$(&HC);CHR$(&H0);    ←28) 000000 001100 001100 000000
0;BCD;CDE;0;       = CHR$(&H0);CHR$(&HE);CHR$(&H1C);CHR$(&H0);   ←29) 000000 001110 011100 000000
0;ABC;FED;0;       = CHR$(&H0);CHR$(&H7);CHR$(&H38);CHR$(&H0);   ←30) 000000 000111 111000 000000
0;AB;EF;0;         = CHR$(&H0);CHR$(&H3);CHR$(&H30);CHR$(&H0);   ←31) 000000 000011 110000 000000
```

Figure 3: Changing Alphabetical Order

PRE-SORT ENCODING

```
FOR X=1 TO LINELENGTH
H=INSTR("abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ", LETTER$(X))
IF H<>0 THEN LETTER$(X)=MID$("badcfehgjilknmporqtsvuxwzyBADCFEHGJILKNMPORQTSVUXWZY",H,1)
NEXT X
```

POST-SORT DECODING

```
FOR X=1 TO LINELENGTH
H=INSTR("badcfehgjilknmporqtsvuxwzyBADCFEHGJILKNMPORQTSVUXWZY", LETTER$(X))
IF H<>0 THEN LETTER$(X)=MID$("abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ",H,1)
NEXT X
```

Figures 2. and 3.

THE CAIRO GENIZA PROJECT

Presenter: Mark R. Cohen *Princeton University*

The texts that form the data-base of the project I am about to describe have a rather interesting provenance. In Judaism it is considered sacrilegious to destroy pages of writing that contain the name of God. When one finds such a loose page or, by extension, any torn or tattered leaf from a sacred book, one should not burn it or throw it away. Instead, one delivers the fragment to a functionary of the synagogue who stores it temporarily on the premises. Normally, when a sufficient quantity of discarded papers has accumulated, or, alternatively, at some regularly appointed time, the accumulation is transferred to a cemetery or some other ground for proper, permanent burial. In some places -- and this is an important but overlooked fact -- -- it was customary to leave the sacred refuse in perpetuity in the synagogue in some sort of chamber designated for that purpose.

The custom I have just described is called "geniza," a Hebrew word meaning both "burial" and the "burial place" for discarded pages of sacred writing. Of the many genizas that have been discovered, one -- the famous Cairo Geniza -- stands out above all the rest. Its uniqueness derives from a happy combination of circumstances. First, the Jews of medieval Egypt used to "bury" not only sacred papers but also secular documents: letters, court records, payment orders -- anything written in Hebrew characters. Most of these documentary writings are written in Judeo-Arabic -- a literary form of medieval vernacular Arabic, which, like Yiddish and other "Jewish languages," is written in the Hebrew alphabet and sprinkled generously with Hebrew words and phrases. Second, unlike the normal case, the Jews of Cairo used a huge, hidden chamber **inside** one of their synagogues as their permanent "geniza." Third, thanks to the dry climate of Egypt, documents dating from as early as the 9th century were found in a fairly readable state when the Cairo Geniza was discovered towards the end of the nineteenth century.

The "secular" fragments, numbering perhaps 15,000, pertain to every conceivable aspect of daily life: economic activities, social organization, family relations, material culture, popular religion, etc. Nothing like it exists for any other Jewish community in the middle ages. Moreover, the Geniza remains are entirely haphazard and uncensored. Since people used the Geniza chamber to discard their personal papers forever, they did not worry about what they might contain. Thus the Geniza constitutes a candid and unmediated source of information -- and not only about the Jews. We find governmental decrees as well as numerous references to the activities of Muslim officials and, occasionally, to Christians, whether they be Oriental, Byzantine, or Latin European. For example, a number of Geniza letters describe firsthand the invasion of Palestine by the Christian crusaders in 1099 and the aftermath of the establishment of the crusader kingdoms in the Levant. More importantly, since the Jews were highly integrated into the surrounding Arab society, what was true of their everyday life was largely true of the Arab majority. Similar documentary finds emanating from the Muslims of the middle ages are extremely rare, so that the Cairo Geniza fills a lacuna in Islamic as well as Jewish history. Equally important, the Judeo-Arabic texts from the Geniza form a priceless source for the history of the Arabic language. Judeo-Arabic reflects a stage of linguistic development that lies in between the Arabic of the Qur'an, which set the standard for all medieval written Arabic, and the medieval (and ultimately modern) colloquial dialects, which are hardly represented at all in either classical or modern Arabic literary sources.

Most of the manuscripts from the Cairo Geniza were dispersed at the end of the nineteenth and the beginning of the twentieth centuries among libraries and private collections throughout the world, from New York and Philadelphia in the West to Jerusalem and Leningrad in the East. Scholars have been hard at work on these collections, studying both the literary fragments and the historical documents, for a century. The historical material, which was undervalued during the early years of Geniza research, has, in the last thirty years or so, occupied center stage. That is largely due to the prodigious textual and historical scholarship of the late Professor S. D. Goitein, whose monumental opus, "A Mediterranean Society: The Jewish Communities of the Arab World as Portrayed in the Documents of the Cairo Geniza ", contains a fascinating and richly detailed account of life during the period from ca. 1000 to ca. 1250, the era most comprehensively represented in the documentary material.

Because of the unsystematic way in which the Geniza manuscripts were dispersed, the tools for research in the field are still relatively under-developed. There exists no unified catalogue of the fragments. One collection, in Oxford's Bodleain Library, was described in the printed catalogue of its Hebrew manuscripts, published in 1906. However, at the time, research on the fragments, and particularly the historical ones, was at a primitive stage. Hence the information in that catalogue about the contents of individual documents is of extremely limited use. Under the supervision of S. D. Goitein and one other Arabist, a useful guide to Geniza documents in all the collections in the world was published by Shaul Shaked under the title "A Tentative Bibliography of Geniza Documents" in 1964. But that catalogue is limited to texts that had been published, in full or in part, and others that had been discussed or quoted in extract form, up to 1963. The project was undertaken in the first instance to direct scholars to the numerous publications on the Geniza documents, dispersed -- much like the manuscripts themselves -- in a myriad of journals and books published around the world and in many different languages.

For the most part, Geniza scholars have always had to rely on their own firsthand perusal of thousands of fragments. This is usually accomplished with the aid of microfilm. Centralized microfilm libraries of the various Geniza collections are housed in New York, Cambridge, England, and Jerusalem. Most scholars, however, would be reluctant to publish a text before traveling to examine the original in the library where it is kept. This is especially true of Geniza historical documents because of the many folds, wrinkles, tears, holes, and dirt spots that mar the letters of the text. Moreover, since scholars are always investigating new problems for which they most likely had not made extensive notes when reading their sources in the past, the conscientious Geniza scholar finds himself faced with the overwhelming task of "re-reading" hundreds if not thousands of fragments with new questions in mind for each new project.

Since the inception of the "computer age," much talk has been heard about applying this relatively new technology for manipulating information to the Geniza. Computerization of a somewhat similar data-base in Hebrew characters, the vast corpus of Jewish responsa literature dating from the early middle ages to the present, has been accomplished successfully at Bar-Ilan University in Israel.

For a variety of reasons, until now the same has not been attempted with the letters and other documents from the Cairo Geniza. For one thing, unlike the responsa, of which there are scores of printed collections, the documentary Geniza materials often consist in intractable manuscripts with formidable paleographic, linguistic, and historical problems. In addition, there seems to be an understandable reluctance to allocate funds for a project that requires the constant attention of people trained in what is essentially a highly specialized and underpopulated field. This is unfortunate since the development of tools and methods for handling the complexities involved in a data-base consisting of Geniza documents would without a doubt be applicable to research on similar groups of historical sources and, at a more general level, to research on any complex textual material.

The advent of IBM's Advanced Educational Project to Princeton, where it has been given the felicitous name "Project Pegasus," provided the stimulus and the opportunity for developing a maiden computer-related Geniza Project at our university. It happens that of the few scholars world wide who are engaged actively in research on the Cairo Geniza documents, two of them are faculty members in the Department of Near Eastern Studies at Princeton (A. L. Udovitch and the present writer), both of whom benefitted from a long-standing collegial relationship with the late Professor Goitein, who spent the final fifteen years of his life and productive career (1970–1985) associated with the nearby Institute for Advanced Study.

Our project, which gained approval in the spring of 1985 and is still in its infancy, has as its goal to teach students how to edit, translate, and exploit for historical research the documentary manuscripts of which I have been speaking. Neither of the two of us who conceived the project had had significant prior experience working with computers. Nonetheless, plunge forward we did, buoyed by the encouragement of university administrators responsible for the entire Project Pegasus and by the friendly and most accommodating staff of the Computing Center.

The proposed data-base to be made available in machine-readable form is the entire corpus of historical documents, both published and unpublished, from the Cairo Geniza. Realistically, it will take an enormous number of man-hours to reach that goal. For the time being, we are designating for computerization documents relating to our respective areas of research: the communal organization of the Jewish communities of the medieval Islamic world (Cohen) and the general economic and social history of Mediterranean Islamic society (Udovitch). The inputting of the data requires laborious typing of texts, some of them previously published and others transcribed in long-hand. Student assistants with good knowledge of Arabic are being employed for this. Use of the Kurzweil Optical Character Reader at the University of Pennsylvania has been investigated but not yet attempted.

The hardware designated for the project is the IBM/XT equipped with Enhanced Graphics Adapter, and the IBM Proprinter. The project calls for the development of tailor–made software and of methods for searching, manipulating, and studying texts in ways not always possible before computers. Because of the inflexibility of commercial text editors for entering and displaying texts in Hebrew and Arabic, it was decided by the Computing Center to develop our own. The product, developed by Michael Sperberg–McQueen, consists of an adaptation of Kedit given the name Kedit/Semitic. Using the IBM Enhanced Graphics Adapter to display Hebrew or Arabic characters on the monochrome screen of the IBM/XT, Kedit/Semitic exploits the keyboard–redefinition facilities of Kedit to assign Hebrew or Arabic characters to keyboard positions, to move the cursor from right to left, to switch from Hebrew or Arabic to English and vice versa, even on the same line of text, and so on. With Kedit/Semitic it is also possible to load the Hebrew and Arabic fonts at the same time and to switch back and forth among the two Semitic alphabets and English during the same editing session. This is a valuable feature, since many Geniza documents are wholly or partly written in Arabic characters; moreover, whenever a non–Jewish "mailman" was the bearer of a Judeo–Arabic letter, the names of the sender and recipient in the address were normally written in Arabic letters for his benefit.

As with the Greek, Latin, and Arabic papyri from antiquity and the early middle ages, as well as the inscriptions on clay tablets or stone from the ancient Near East, the Geniza documents pose many paleographical problems that must be taken into consideration during the process of entering them into the computer's memory. For instance, we must be careful to employ proper critical apparatus –– dots, parentheses, slashes, and square brackets –– to signify lacunae, deletions, insertions, and other peculiarities of the texts, as well as completions suggested by the editor. In addition, in order for entered texts to represent faithfully the orthography of the original manuscripts, we have to follow the practice in Judeo–Arabic Geniza documents of employing diacritical marks with absolute lack of consistency. Thus, a Hebrew consonant representing a letter that in Arabic carries a diacritical sign, will appear both with and without the normal Judeo–Arabic equivalent of that mark in one and the same manuscript. Of course, we must also be able to command the system to ignore such encumbrances as dots, parentheses, square brackets, and slashes when searching for words, terms, phrases, names, and other relevant character strings. These capabilities should be attainable with the proper data–base management system. For the time being, we are examining available PC concordance programs, such as the one developed at Brigham Young University.

Down the line we will develop data–base systems to handle the complex interrelationships of text, paleographic information, textual–critical commentary, linguistic analysis, historical interpretation, and bibliographic references and to permit sophisticated searches for information relevant to particular research topics. Searches for individual word forms, roots, syntactic locutions, or semantic fields, should be possible with the right data–base structure. It should also be possible to reunite fragments of texts and thereby achieve the same level of satisfaction that Assyriologists derive from making "joins" between broken pieces of the same inscription.

Once the software to perform these sophisticated functions has been developed and we have entered a significantly large number of texts into the data–base, it will be possible for students to use the computer to do meaningful research on specific historical problems. For example, a student working on family, kinship, and social organization would employ key search words or phrases, such as the terms in Arabic or in Hebrew for brother, son, brother–in–law, son–in–law, uncle, aunt, cousin, etc. He or she would also search the Geniza data–base for names of known relatives. After retrieving the relevant documents or passages within documents, the student would then proceed to ask of the collected material important questions, such as, what are the notions of kinship, under what circumstances are kinship terms used beyond precise kinship meaning, and what are the real categories in which people thought about close relationships. To facilitate research, the students will be able to call up files containing important bibliographic, linguistic, and historical information, and in some cases, complete English translations of particular texts.

Right now, at this still early stage of the project, we are using the computer mainly to teach the transcription, editing, annotation, and translation of Judeo–Arabic documents. The admission requirement of our seminar is a good working knowledge of classical Arabic. Hebrew is optional, and for the Arabist who knows no Hebrew, learning its alphabet is relatively simple, and, indeed, a welcome change from the struggle to master the more difficult orthography of Arabic. The student is presented at the outset with a photocopy of a relatively easily decipherable manuscript. Since legal documents, with their inherent need for clarity, tend to be written in a style of Arabic closely approximating the classical Arabic with which the student is familiar, we choose one of these for starters.

A two to three hour tutorial on using the IBM/XT and the Kedit/Semitic software proves to constitute a sufficient introduction to the technical side of the seminar for the students to begin independent work. My office sits directly opposite the "student cluster" of four PCs and an IBM Proprinter, and I encourage students to knock on my door for assistance whenever it is needed, no matter how trifling the problem might seem. We have found that our bright students have many valuable ideas about procedure. At the suggestion of a student, for instance, it was decided to create for each document a "help" file including linguistic comments and, for the benefit of the student who does not know Hebrew, an English translation of those Hebrew or Aramaic words and phrases that invariably crop up in Judeo-Arabic texts.

Between meetings of the seminar the student works independently in the computer room, deciphering an assigned text and entering it into the computer. Periodically I check on the progress being made by examining on screen the student's computerized transcription. Using the new "multview" feature of Kedit (version 3.52), I type comments and corrections at the bottom of the file. Since Kedit/Semitic allows us to work in both Semitic alphabets during the same editing session, I am able to provide Arabic transcriptions of Judeo-Arabic words whose orthography is irregular. When the student returns to the computer room to continue transcribing the manuscript, my notes at the bottom of the document, displayed on a separate screen, provide individualized assistance in recognizing and understanding mistakes, and ultimately, in comprehending the text and its significance. At each seminar meeting we hear the student's edition and translation of the text under consideration and hold a discussion of relevant paleographic and linguistic features, as well as the document's historical setting and significance.

As I said earlier in this paper, our project is still in its infancy. We cannot begin to ask our students to engage in actual research until we have compiled a large enough data-base. During the next year we hope to muster a corps of assistants to enter texts, with a view towards expanding the tasks of the seminar. We look forward to the day when our data will outgrow the ten megabytes of the XTs hard disk, and we will have to store the data in the university's main-frame computer, with which our PCs will then have to be linked. However, even now, at this preliminary stage, I see the advantages of using the computer. It is its own motivational device. It sweetens the laborious task of reading and transcribing difficult texts. It allows for individualized instruction without concern for conflicting schedules. In the words of a student who, in a very few weeks, has acquired significant experience with manuscripts she had never before seen, let alone tried to decipher, and, in addition, has had her very first experience using a personal computer, preparation for the seminar is like "playing with a toy." And, as a method of learning to handle extremely challenging and difficult texts, "it really works."

WORKSTATIONS FOR COMPUTER MUSIC AT CARNEGIE MELLON

Presenter: Roger B. Dannenberg *Carnegie–Mellon University*

copyright © Roger B. Dannenberg, 1986

ABSTRACT

Computers provide many opportunities for improving music education, for creating music, and for music research. Unfortunately, almost all computer music applications must be built starting at a very low level. Special purpose hardware must be interfaced for sound production, real–time software must be developed, and music notation tools must be created. Two systems are under development at Carnegie Mellon to support music applications of computers by providing developers with a set of music–oriented software tools. The first of these, the CMU MIDI Toolkit, using low–cost, off–the–shelf hardware, is already in everyday use by students and will be available for distribution in the near future. The second system, the Musician's Workbench is a more ambitious project using advanced personal computer technology, and promises to provide a powerful facility on which to build a wide range of educational applications.

1. INTRODUCTION

Computer music education, composition and research efforts have long been hindered by the requirement of high–quality audio output. While this technology has been with us for quite some time, practical implementations have historically been expensive both in terms of equipment and human resources. A further problem has been that most implementations are hardware dependent, forcing new effort every time a computer system is upgraded. A number of computer music groups have opted to hold onto obsolete computing equipment rather than face a new effort to construct audio output facilities for an up–to–date computer system. Since few groups can afford to produce many copies of a hardware system, classroom music instruction with individual student access to computer systems is rare.

This problem is greatly reduced by the recent efforts of manufacturers to produce low–cost synthesizers and computer interfaces based on a standard protocol called MIDI [1]. Given this low–cost and readily available hardware, it is now relatively simple to obtain audio output from a personal computer. For computers without commercially available MIDI interfaces, new VLSI chips and intelligent MIDI processors are available to simplify MIDI interface construction.

This approach has several important implications for music education. First, educators can now expect interesting timbres rather than unmusical beeps in their music-oriented programs. Secondly, musicians are now able to work with real-time synthesis. In other words, sound can be obtained instantly and can be modified as it is being produced. This is in contrast to early computer music production systems that require a lengthy preparation step before sound can be heard. Real-time capabilities have led to studies of performance and the construction of interactive computer music programs. Finally, the lowering of costs has enabled many schools and even individuals to equip themselves with advanced synthesis and computing facilities.

An effort is currently underway at Carnegie Mellon to provide computer systems to assist musicians, composers, researchers, and music educators. For several years, we have been using a text- and programming-oriented system for teaching Computer and Electronic Music I and II, courses that emphasize hands-on studio experience. The system began with custom-built hardware and software. Now in its third generation, the system uses off-the-shelf low-cost hardware and a software suite written at Carnegie Mellon called the CMU MIDI Toolkit. This software will soon be available, and is described in greater detail in Section 2.

While the CMU MIDI Toolkit serves many of our needs, it lacks features we would like for general music applications. In particular, a screen-oriented interface that displays music in traditional notation is an essential component of many music application programs. Such an editor can ease the task of producing printed scores and copying parts. Moreover, it can provide a foundation for educational programs to teach music theory, sight-singing, and orchestration.

We are working on such an editor as part of an advanced computer music system called the Musician's Workbench. At this time, a data structure for supporting the Musician's Workbench editor has been implemented, and a very simple editor has been constructed for testing and evaluation. Section 3 describes the current status of this project and some of our plans for the future.

2. THE CMU MIDI TOOLKIT

Because the Musician's Workbench is such an ambitious project, we have been forced to develop an interim system for immediate use in our studio at Carnegie Mellon. This system is called the CMU MIDI Toolkit. I will first describe some of the functionality of the system and then discuss some of its current applications.

2.1. Adagio: One of the central components of the CMU MIDI Toolkit is a language, Adagio, which provides a text representation for a note-level view of a musical score. A few lines of an Adagio score serve to illustrate the essential features of the language:

```
!tempo 100
A4 Q Lmp V3
C H Z15
TW2 CS4 U58
```

The first line establishes the tempo in beats per minute. (Tempo may be changed at any point, and multiple overlapping tempi are possible.) The second line describes a note. In Adagio, a note is described on one line which contains a collection of attributes separated by blanks. Attributes may occur in any order. In the example above, the first attribute, A4, means that the pitch is the fourth-octave A (nominally A440). The Q means the duration is a quarter (one beat), and Lmp means the loudness is mezzo-piano. Finally, V3 means to play the note on MIDI channel 3.

The second line specifies a note whose pitch is C. Since no octave is specified, the C nearest the previous note, C5, is chosen. The duration is a half note, and Z15 means to choose timbre number 15. (The meaning of 15 is dependent upon the synthesizer used to perform the score.) By default, this note will start immediately after the previous quarter note and be played on channel 3. The loudness will again be mezzo-piano. In general, attributes that are not explicitly stated are inherited from the previous line.

The last line specifies a note that starts at a time that is the duration of two whole notes from the beginning, thus TW2. Time could also be specified in hundredths of a second; for example, T400 means start at time 4 seconds. This note has a pitch of C-sharp above middle C (CS4), and the duration is specified as 58/100 seconds.

One of the problems often encountered with MIDI software is the notion that music has a rock-steady tempo and that scales should be equal-tempered. I am unaware of any commercial software that supports poly-rhythms, multiple tempi, or that allows time to be specified in seconds rather than in beats. This bias toward traditional music may be appropriate for commercial music, but is not appropriate for exploring experimental musical concepts and music theory at a university.

The CMU MIDI Toolkit and Adagio try to avoid the traditional music biases by reducing all metrical information to time measured in hundredths of seconds. The composer is still free to think in traditional terms, as durations can be expressed in terms of beats as well as seconds.

In the area of pitch, we are faced with a serious deficiency in MIDI, which is that MIDI encodes pitch as a small integer that refers to a scale step, not to an arbitrary frequency. One way to overcome this problem is to use a second MIDI function called pitch bend to alter the tuning of a synthesizer for each note. Pitch bend allows very fine control over pitch and with computer control, it allows the production of any desired tuning. Adagio will soon provide an option where each pitch can be translated into a pair [MIDI pitch code, MIDI pitch-bend code] to give precise control over the synthesizer. The Adagio composer will then be able to define his or her own scales and even play the same piece of music with different tuning specifications.

Another problem with commercial software programs is that they are often difficult to use as components in an integrated software system. This is partly because these programs are conceived of as closed systems; the only interfaces are the one with the user and MIDI. Software interfaces are not provided, nor is source code available for these programs. Consequently, making modifications to suit a particular environment or application is impossible. The CMU MIDI Toolkit is designed from the beginning to be modular and flexible. For example, the Adagio program has a "front-end" that translates Adagio scores into an internal form and a "back-end" that controls a synthesizer accordingly. I have rewritten the back-end on several occasions to drive different devices and to produce and format input for other programs. Adagio itself is an interesting software interface. Several students have written composing programs that produce Adagio scores. Since it is often easier to output an Adagio score than it is to control a synthesizer in real time, Adagio is an effective way to interface a composing program to a synthesizer.

2.2. Transcribe: Another component of the CMU MIDI Toolkit is Transcribe, a program for recording keyboard performances and encoding them as an Adagio programs. Transcribe is interesting because it produces a readable text output which can be studied, subjected to computer analysis, or edited as text in order to correct mistakes or perform some transformation. The function of Adagio and Transcribe are combined in the program Record, which can record a keyboard performance while simultaneously playing an Adagio score. This allows complex scores to be entered one part at a time.

2.3. Moxc: Last but not least, the CMU MIDI Toolkit contains software for controlling synthesizers directly from C, a popular algorithmic programming language. Thus, programmers can build their own tools for music research without worrying about low-level details of MIDI.

For more advanced programmers, a C language extension called Moxc allows students to write real-time programs that interact with musicians. Moxc is a C implementation of Moxie [2], a language developed by Doug Collinge at the University of Victoria. The principle feature of Moxc is the routine "cause", which schedules an action to take place in the future. For example,

```
cause(25, note, 48);
```

will cause the parameter 48 to be passed to the routine note after a delay of 25 hundredths of a second.

2.4. Current Use: The CMU MIDI Toolkit is an open-ended system designed more to facilitate work in computer music than to teach any particular skill. It is used in two courses at Carnegie Mellon, Electronic and Computer Music I and II. The goal of these courses is to teach the theory and techniques used to produce experimental electronic and computer music. The course emphasizes active learning by doing in our studio, and the CMU MIDI Toolkit provides a flexible system that enables students to learn one step at a time.

As a first step, Adagio can be learned very quickly. This is important since it provides many students with their first introduction computing. These students must learn so many concepts at once (editing, files, commands), that the familiar musical language of Adagio is a great asset. Later, students discover and begin to explore the full range of possibilities Adagio provides.

At that point, some programming concepts are introduced and related to music. For example, a sequence of statements in a programming language is like a sequence of notes (a melody), and a loop in a programming language is like a music repeat. A set of "include" files enables students to write simple programs in the C programming language whose output is music.

Once simple control constructs are mastered, students want to know how to produce music with chords and polyphony (multiple notes sounding simultaneously). Moxc provides this capability and can be introduced as a fairly simple extension to what is already understood. To illustrate how Moxc is used, one student wrote an "echo" simulation program that produces rapid sequences of notes that sound like echos in response to a keyboard input from a synthesizer. The speed and decay rate of the echoes can be adjusted. The CMU MIDI Toolkit is also being used to explore tuning systems, to write music, and for some studies in music performance.

The CMU MIDI Toolkit runs on an IBM-XT or AT with a MIDI interface manufactured by the Roland Corporation (the MPU-401 with IBM-PC interface adapter). The released version of the software will include source code and will require a Microsoft C compiler, except for the Adagio, Transcribe, and Record programs. Since MIDI is a reasonably standardized interface, many synthesizers will work with the software. We use a Yamaha DX7 primarily for its keyboard input, and a Yamaha TX816 for synthesis. I will use a Yamaha DX100 at the conference presentations.

3. THE MUSICIAN'S WORKBENCH

The CMU MIDI Toolkit is meant to be an interim solution to specific problems, and it certainly does not address all of the needs of music educators. To this end, we are developing an advanced system called the Musician's Workbench, which will consist of a powerful personal computer, special purpose hardware for music synthesis, and graphics-based software for music.

3.1. Musician's Workbench: Hardware For music synthesis, we are cooperating with Dr. Peter Comerford at the University of Bradford, England, in an effort to interface his Bradford Musical Instrument Simulator [3] to a computer workstation. The Bradford Musical Instrument Simulator is a very high quality music synthesis device that performs about 20 million floating point operations per second and will be capable of realistically simulating a chamber orchestra. This device is in the early stages of optimization, and we believe this performance could be achieved with 20 to 30 chips using currently available VLSI technology.

Because commercial music keyboards are available at low–cost, we will use MIDI to provide keyboard input to the Musician's Workbench.

The music hardware will be interfaced to a personal computer (an IBM RT is a candidate) with a large bit–mapped display, virtual memory, a network interface, and support for real–time control. We are currently using the CMU Andrew system for development work and plan for the Musician's Workbench to be an extension of Andrew.

3.2. Musician's Workbench Software: The main task in the Musician's Workbench project is to design and implement software for displaying, editing, performing, and printing music. The software must be flexible enough so as not to preclude its use in music education, experimental and electronic music, and applications we have not yet envisioned. We have begun by building an extensible data structure for music representation, and a very simple editor has been implemented for testing purposes. The data structure provides the following features:

– Events. A piece of music, or score is a set of events, which may be created, deleted, or modified.

– Properties. An event has a list of associated properties, each of which has an attribute or name, and a value.

– Hierarchies. Events can be related in multiple hierarchical structures.

– Views. An event can participate in any number of views, which are subsets of the score. Events may have view–specific properties.

– Incremental Update. A view can be associated with one or more regions of a computer display. When visible events of a view are modified, mechanisms provided by the data structure allow view software to modify the display incrementally.

– Undo. The data structure uses a history mechanism to enable previous versions of the data structure to be restored. This undo facility is completely integrated with the incremental update mechanism so that changes can be rapidly "undone".

– Permanent Storage. Scores can be written to a file and read back into memory. Circular data structures and shared structures are properly handled. Assuming music editing exhibits locality of reference, then saving and restoring a score will, as a side effect, rearrange its placement in virtual memory so as to optimize paging performance.

– Event Sets. Events can be collected into sets (implemented as hierarchies). Abstract operations exist for creating and manipulating event sets. For example, one operation creates a set containing every event such that the value of a given property is greater than a given amount. Set union and intersection are also implemented.

– Editing Operations. A collection of abstract editing operations is provided. For example, one operation takes every event in a given set and increments the value of a given property by a given amount.

All of this functionality is implemented and will be demonstrated at the conference. The current music editor provides a "piano roll" notation where pitch is indicated by position on a grand staff, and time is indicated by horizontal position. A note is indicated by drawing a heavy horizontal line from the note's starting time to its ending time at the appropriate vertical position corresponding to the note's pitch.

The next stage of development is the creation of a set of music fonts and the development of a representation for traditional Common Music Notation within the existing data structure. This work is currently in progress.

4. CONCLUSIONS

The CMU MIDI Toolkit provides an easy–to–use software interface to produce music, and it runs on inexpensive hardware. It can be used to teach students about sound and music, or as a base upon which to develop software oriented toward a specific educational objective. The original version of the software has been used successfully to teach children as young as nine years how to use computers to create music [4]. The current version is also a powerful tool for composition, music production, and research.

Current effort is being applied to the development of an advanced computer music workstation, the Musician's Workbench. Among our objectives are to (1) allow composers to produce orchestral-size scores in publishable form, (2) to realistically simulate a chamber orchestra in real time, and (3) to provide a suitable foundation for computer music education, composition, and research.

5. ACKNOWLEDGEMENTS

Carnegie Mellon has provided the stimulating environment without which none of this work could have been accomplished, and many people have contributed and/or led the way. Chief among these are Paul McAvinney and Marilyn Taft Thomas. Paul helped formulate the concept of the Musician's Workbench, built our two copies of the Bradford Musical Instrument Simulator, and has worked long hours to turn the Musician's Workbench into reality. Marilyn used her rare combination of organizational and artistic abilities to create the CMU Computer Music Studio and its curriculum, which in turn drove the design of the CMU MIDI Toolkit. Phil Miller showed me how to teach programming to musicians, and this also had a beneficial impact on the software design. The CMU MIDI Toolkit contains software contributions from Dean Rubine, Joe Newcomer, and Rob Chandhok.

REFERENCES

[1] Gareth Loy. Musicians Make a Standard: The MIDI Phenomenon. Computer Music Journal 9(4):8–26, Winter, 1985.

[2] D. J. Collinge. MOXIE: A Language for Computer Music Performance. In Proceedings of the International Computer Music Conference, pages 217–220. Computer Music Association, 1984.

[3] P. J. Comerford. Bradford musical instrument simulator. IEE Proc. 128, Pt. A(5):364–372, July, 1981.

[4] Frances K. Dannenberg, Roger B. Dannenberg, and Philip Miller. Teaching Programming to Musicians. In 1984 Proceedings of the Fourth Annual Symposium on Small Computers in the Arts, pages 114–122. IEEE, October, 1984.

INDEX BY PRESENTER / AUTHOR, VOLUME II

Presenter / Author **Page**

Beauchamp, James 60
Becker, Donald A. 91
Causey, Robert L. 64
Cohen, Mark R. 101
Hill, John W. 45
Horan, Fred 11
Hunter, Scott 11
Kokkinos, Chris 11
Lidell, Gail 11
Mackenzie, Arch 11
Martin, Robert L. 83
McGraw, Deborah 32
Niesz, Anthony J. 48
Noblitt, J. S. 36
Owen, David I. 2
Paperno, Slava 32
Pet, W. J. A. 36
Pinkston, Russell 78
Sol, D. F. 36
Sutter, Carl 11

INDEX BY UNIVERSITY, VOLUME II

University	Page
Cornell University	2 11 32 36
Princeton University	101
University of California, Los Angeles	83
University of Illinois	45 60
University of Texas	78
University of Texas at Austin	64
University of Wisconsin, Madison	91
Yale University	48

IBM

Academic Information
Systems Proceedings

1986 University AEP Conference

Volume III:

PHYSICAL AND LIFE SCIENCES

April 5 – 8, 1986
Sheraton Harbor Island Hotel
San Diego, California

CONTENTS, VOLUME III

Title **Page**

An Intelligent Chemical Stencil for 3D Molecular Coordinates 2

Calculus Software From the PC to the RT 10

Computer Aided Instruction in Neuroanatomy 16

Computer Graphics for an Intermediate-Level Optics Course 29

Computer Graphics Utilities in Special Relativity 41

Computer Simulations of Plant Morphology and Evolution 65

Crystal 72

Development of a Computerized Practice Simulation ....

Dynamic Models in Biochemistry: Using Spreadsheets 81

Evaluation of a Bayesian Forecasting Program ....

Intelligent CIA 93

Interactive Graphics/Display & Analysis/Earthquake Seismograms 109

IRIS: A Tool for Learning Data Analysis Strategies 114

Molecular Structure Graphics for Undergrad Chemistry Ed. 120

New Elementary Physics Laboratory Experiments 124

Representation of Density Distributions on a Sphere ....

The Use of Expert Systems to Teach Veterinary Decision-Making 142

Using CAMPS to Teach Medical Problem Solving 147

INDEX by Presenter/Author, Volume III 158

INDEX by University, Volume III... 160

AN INTELLIGENT CHEMICAL STENCIL FOR CALCULATION OF 3D MOLECULAR COORDINATES

Presenter: Michael B. Bolger *University of Southern California*

copyright © University of Southern California, 1986.

GOAL

Education in Pharmacy at the chemical level involves teaching students to recognize chemical structures and to correlate biological activity with the molecular functional groups found in a given drug. Interactive three-dimensional computer graphics allows one to visualize and quickly grasp the important aspects of stereochemistry and conformational isosterism that exist between synthetic drugs and corresponding endogenous chemicals found in the body. One example of this involves the molecular similarity between dopamine (an endogenous neurotransmitter) and haloperidol (a neuroleptic used to treat schizophrenia). At first glance, there seems to be no similarity between the two structures. Historically, students have had a very hard time under- standing how the synthetic drug molecule can have a blocking action at dopamine receptors. Refer to Figure 1.

The advantage of a computer-assisted molecular graphics system has been previously demonstrated in a classroom situation by using the portable PC with an RGB video projector and generating stereo pair images in two different colors so the stereo can be observed with 3D glasses. This mode of teaching allows one to reinforce concepts with a very strong visual picture, in addition to the didactic message that neuroleptics are antagonists at the dopamine receptor. There is no other visual aid which can match molecular graphics for accuracy and impact in teaching chemistry.

COMPUTING STRATEGY

We have previously developed and reported on a program called MOLDISP.EXE for stereo display, rotation, scaling, and animated motion of molecules up to 350 atoms. These friendly, interactive programs were designed to be used by the instructor for classroom demonstrations and to generate images which can be merged into CAI modules. This program is able to display and manipulate two molecules simultaneously as well as to perform bond rotations, and orientation of a selected bond on the X-axis. The latter feature is useful when molecular structures are compared for stereochemical similarity.

These software tools are only as good as the database of 3D-coordinates that are available for display. The problem in the past has been the lack of X-ray crystallographic structures that are available for classroom demonstrations and computer assisted instruction. There are basically three ways of obtaining 3D molecular coordinates. First, one can look up the coordinates in reference volumes and type them into a connectivity table manually. Secondly, one can download the coordinates from a vendor of chemical information such as CIS. These two methods have the disadvantage of only being available for molecules which have been crystallized and whose structure has been determined. From a teaching point of view, many other molecules would be of interest such as solvents which cannot be crystallized. The final option is to develop the three dimensional coordinates by calculation from chemical principles. This is the solution that has been addressed by ChemCil.EXE.

TOOLS

A new program has been developed which will allow the user to "draw" a molecule on the video display monitor with a mouse-assisted, intelligent "chemical stencil". The molecule is drawn in two dimensions by the user and a connectivity table is generated. Refer to Figure 2.

When the molecule is finished, a three dimensional model can be calculated by using an extensive set of standard bond angles and bond lengths. When in doubt the program will prompt the user for dihedral angles. This method can be used for any chemical that the instructor can imagine including fused ring structures and molecules which may not even exist yet. Also, this method will allow the instructor to illustrate the steric consequences of chemically unstable geometric conformations and generate stereo isomers which can be displayed and rotated in perspective to illustrate the axis of symmetry.

Technically, this project was more complicated than the display routines. The roots of the program were developed in the summer of 1984 when most of the screen graphics and mouse drivers were written. The most challenging aspect has been to impart a modicum of chemical intelligence to the program. This program is definitely not an example of artificial intelligence. However, it certainly knows more chemistry than the average 3rd year pharmacy student. For example, one algorithm was developed for determining the appropriate valence of an atom which has been draw on the screen and then for warning the user if too many bonds are drawn. This type of calculation can be used in another subroutine to add the correct number of hydrogens to a molecular connectivity table in order to fill out the appropriate valence. Figure 3 shows the data structure which is found in ChemCil. The structure consists of a connectivity table and X, Y, & Z cartesian coordinates.

A short and powerful algorithm was developed to calculate the valence of an atom from the test string which contains the atom type ie "C " for carbon. The valence for carbon is 4 and there are several types of geometry which can occur as a result of specific bonding properties. The input to the following algorithm is ID = the index number of the current atom. Outputs from this program are NVAL = the valence of the the atom and IDCON = two times the number of identical connections to other atoms from the current atom. For example, if the molecule is ethene as shown above, carbon 1 is single bonded to hydrogens and double bonded to carbon 2. Therefore, the valence of carbon 1 would be 4 and the number of identical connections would be 2. The reason the variable IDCON is equal to 4 is that when counting identical connections the program counts the connection that is being compared as well as other connections resulting a value which is always double the actual number of identical connections.

```
ATOM$ = "C N P O S H F ClBrI "
```

Call to Subroutine: ID = index number of current atom

```
27500 '------- Subroutine to Count Identical Connections ---------
27510  NATOMPOS = INSTR(1,ATOM$,A$(ID))
27520 IF NATOMPOS = 1 THEN NVAL=4: GOTO 27550
27530 IF NATOMPOS < 6 THEN NVAL=3: GOTO 27550
27540 IF NATOMPOS <10 THEN NVAL=2: GOTO 27550 ELSE NVAL=1
27550 NCON=0
27560 FOR KK=4 TO 7
27570         FOR  LL=4 TO 7
27580         IF MOL(ID,KK) = 0 THEN 27610
27590         IF MOL(ID,KK) = MOL(ID,LL) THEN NCON=NCON+1
27600         NEXT LL
27610 NEXT KK
```

```
27620 IDCON = NCON-NVAL
27630 RETURN
27640'
27650'
```

At this point the program can branch as desired with the following:

```
ON IDCON/2  GOTO  XXX,          YYY,      ZZZ,    AAA
                  tetrahedral   trigonal  linear  linear
```

The outputs of this algorithm can be interpreted in terms of molecular geometry as follows:

Atom Type	Valence	Identical Conn.	Geometry	Chemistry
C		IDCON = 6	Linear	Acetylene
	NVAL = 4	IDCON = 4	Linear	Allene
Si		IDCON = 2	Trigonal	Aromaticor C=C
		IDCON = 0	tetrahedra	Saturated
N				
		IDCON = 4	Linear	Nitrile
P	NVAL = 3	IDCON = 2	Trigonal	Aromatic or N=
		IDCON = 0	Tetrahedral	Amine
As				
O		IDCON = 2	Trigonal	Carbonyl
	NVAL = 2	IDCON = 0	Tetrahedral	Ether
S				
Halogens & H	NVAL = 1	IDCON = 0	Single Bonds	

SUMMARY

Several mainframe programs are available, without graphics and with no intelligence, for calculating molecular geometries from user supplied bond angles, bond lengths, and dihedral angles. However, if the user has a simple tetrahedral molecule with four atoms the calculation fails because there needs to be at least four atoms connected in sequence to define the dihedral angle.

In addition to large molecules such as those above, ChemCil is able to recognize such molecules as diatomics, triatomics, and simple tetrahedral molecules and calculate the appropriate geometry automatically. Rings are a problem for any molecular modeling program and ChemCil must prompt the user with its best guess for bond angles and dihedral angles in order to be sure that the rings will close properly.

EDUCATIONAL RESULTS

Student reaction to molecular graphics in the lecture hall has been enthusiastically positive. Although we do not have separate evaluations for the molecular graphics parts of the course, a general measure of student evaluation would indicate that this project has improved the educational experience. This project must now move on toward implementation of molecular graphics into CAI. A pilot demonstration of this type of CAI was presented at the AEP conference in 1985. The pilot was disappointing in that the authoring system that was used was not flexible enough to allow for animated motion of molecules and stereo displays on a time frame that is appropriately paced for learning. These problems were unsolved at the start of the 1986 AEP conference, but as I am writing this on Monday afternoon, I have found the solution in the form of ScriptWriter.

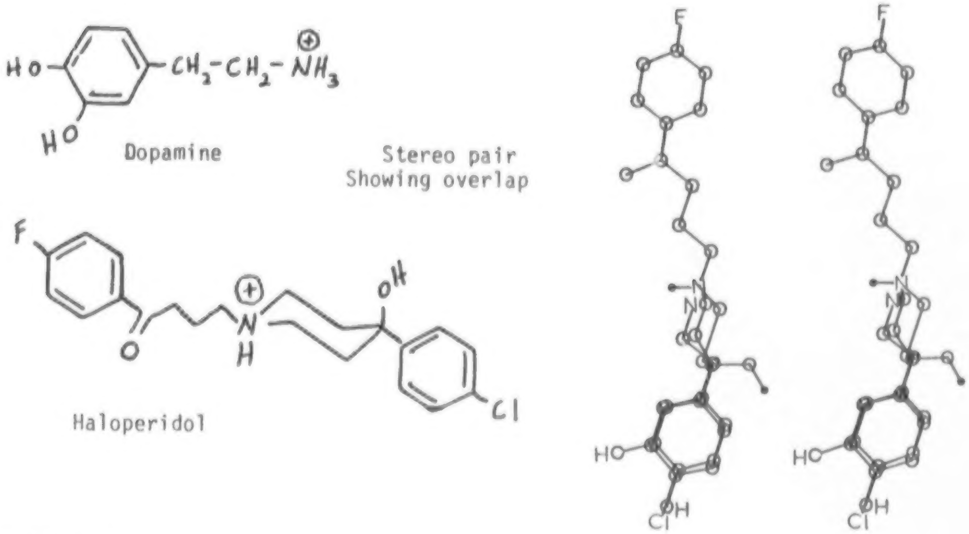
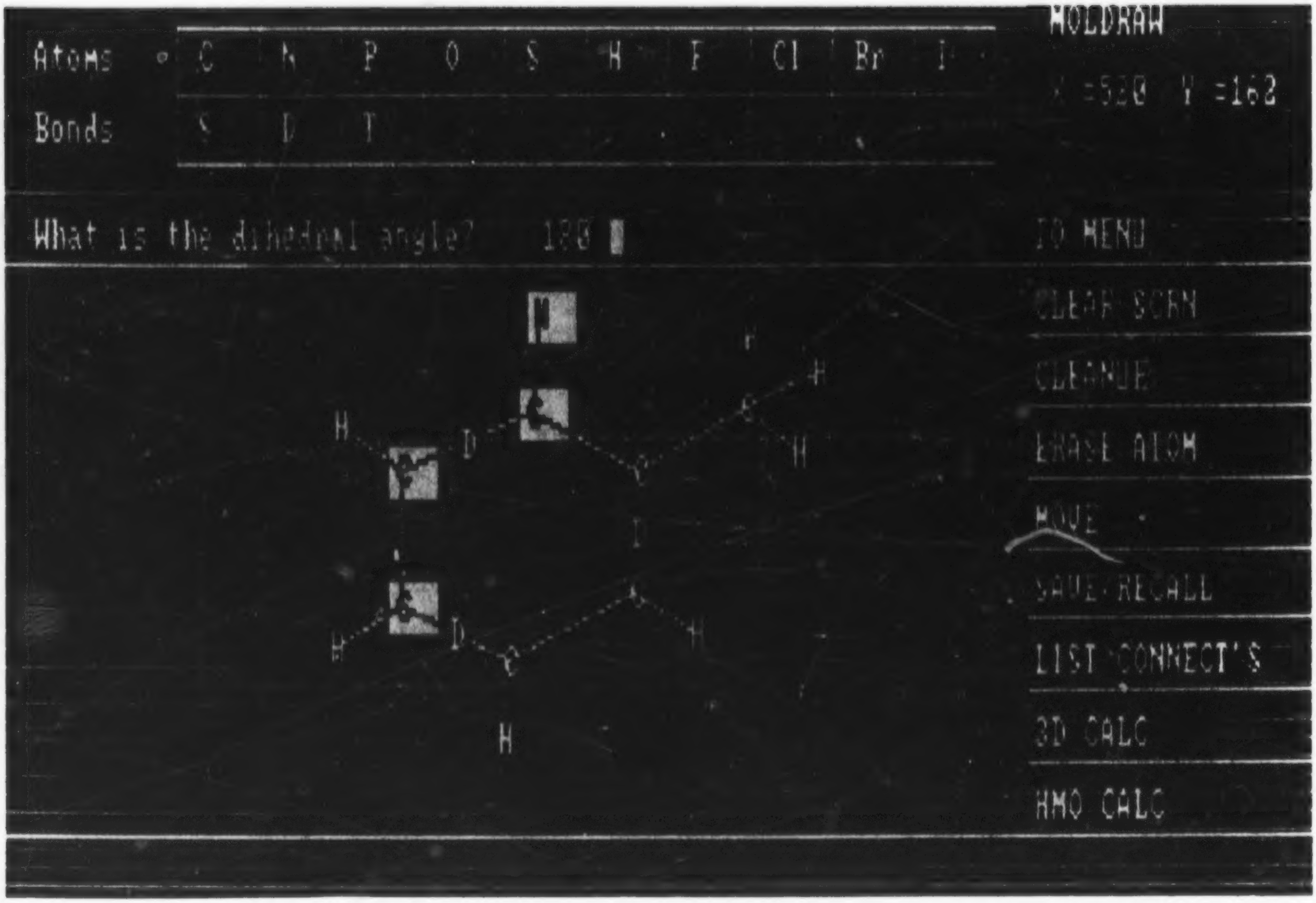
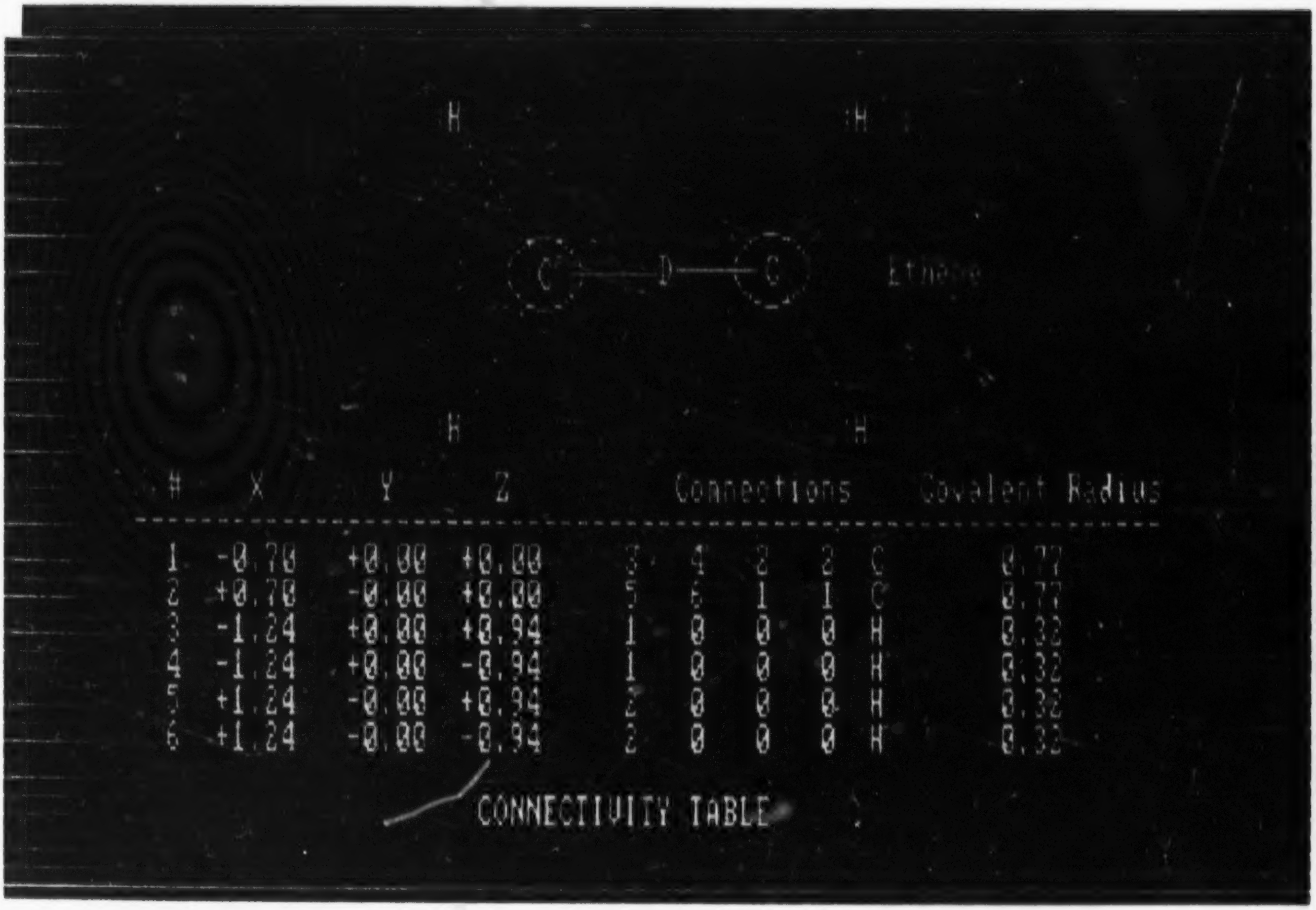


Figure 1.
Stereo Pairs



What is the dihedral angle? 180

Figure 2.
Chemcil Drawing Screen



```
 #    X        Y      Z            Connections      Covalent Radius
------------------------------------------------------------------------
 1  -0.70   +0.00  +0.00       3   4   2   2  C          0.77
 2  +0.70   -0.00  +0.00       5   6   1   1  C          0.77
 3  -1.24   +0.00  +0.94       1   0   0   0  H          0.32
 4  -1.24   +0.00  -0.94       1   0   0   0  H          0.32
 5  +1.24   -0.00  +0.94       2   0   0   0  H          0.32
 6  +1.24   -0.00  -0.94       2   0   0   0  H          0.32

                    CONNECTIVITY TABLE
```

Figure 3.
Connectivity Data Structure

CALCULUS SOFTWARE FROM THE PC TO THE RT: CMU TUTOR AND ANDREW FROM A PC USER'S VIEWPOINT

Presenter: Russell C. Walker *Carnegie–Mellon University*

My first effort at software to accompany a calculus sequence was written in BASIC to run on the IBM PC equipped with a Hercules Graphics Card. This report will discuss the first steps in converting my materials to run on the IBM RT in the ANDREW environment using CMU Tutor as well as the expected improvements that will be possible when ANDREW is fully developed.

EDUCATIONAL OBJECTIVE

Many calculus concepts are highly geometric, and their understanding is often facilitated by careful graphic illustration. The objective of this software is to enhance understanding by providing demonstrations which the student can generate and repeat. Particular emphasis is given to topics with which students frequently have difficulty, such as the limit and the remainder theorem for Taylor polynomials.

Topics treated include: The definition of the limit, graphing and root determination, the definite integral and applications, Archimedes' approximation to pi, numerical integration, approximation by Lagrange polynomials, the Fundamental Theorem of the Calculus, Taylor polynomials, Fourier series, differential equations, the definitions of the trigonometric functions, harmonic motion, graphing in polar coordinates, and graphing of surfaces.

THE ENVIRONMENT OF CMU TUTOR

The CMU Tutor language was developed to enable persons with little programming experience to develop educational applications without having to depend completely upon professional programmers. To achieve this goal, the ANDREW resources have been employed to produce a development system featuring helpful diagnostics, quick program testing and modification, on-line help, and a powerful editor.

Tutor programs are developed in a two part window. The source code appears in the upper half within a text editor and the results of executing the program appear in a display window in the lower half. Thus, one may examine the code while observing its output.

Between the two portions of the window is a section listing a number of the most commonly used commands. These commands may be inserted into the program by placing the cursor at the proper position in the file and clicking the mouse on the desired command. Screen locations can also be entered where required by clicking the mouse at the desired location in the display window.

Extensive help is available in the form of a language manual which appears upon request in a second window. A scroll bar and a list of main headings indicate topics on which help is available. Help on a specific topic is requested by means of a mouse click on the topic. Within the help file itself, searching is possible by the same means as in the source code window. Sample segments of code provided in the help file may easily be copied into the user's program.

Figure 1 shows the development mode of CMU Tutor. The help window shows help on the commands used to designate the origin of a set of axes.

File level operations such as execution, saving, searching, getting help, etc. are initiated by means of menus controlled by the mouse.

Upon execution, syntax is checked, the cursor moved to the location of the error, and an error message – usually quite helpful – is provided. Execution can be resumed when the error has been corrected. CMU Tutor owes much of the speed of its test and revise cycle to incremental compiles – when the source code is changed, only the changed portion is recompiled.

COMPARISON WITH BASIC ON THE PC

It would be difficult to find an area of program development in which CMU Tutor would not prove superior to BASIC, and when the project under consideration is instructional software, I suspect that the same could be said for most languages not specifically designed for that purpose.

The most immediately obvious advantage is in the presentation of text. The inclusion of text formatting commands within the editor makes features such as justification, centering, and font selection elementary. Text and graphics displays mix freely.

CMU Tutor is a structured language, forcing the successful programmer to adopt a modular approach with a controlling master module.

Line numbers are not used, which is a boon except for those addicted to the GOTO. Labels to allow implementation of a GOTO are not currently implemented.

Tutor will accept student entered functions using usual mathematical notation rather than requiring that functions be entered using notation acceptable to BASIC. Further, error routines to test student entered functions for proper syntax are easily implemented.

A special advantage to Tutor is the inclusion of commands to judge student responses. The programmer needs only to set up the criteria that characterize a correct or incorrect answer, and Tutor carries out the details of deciding the correctness of a response. The parsing of an expression required of the BASIC programmer is thus done by Tutor.

The screen size limitation of the PC largely disappears on the RT workstation. On the PC, when a graph of any size is on the screen, there is little room for other information. The RT display will hold roughly the same amount of information as six PC screens.

A potential disadvantage is the freedom that ANDREW gives to the student in running a program. The user can request a window of virtually any size when running a program, and this forces the programmer to write flexible materials. Fortunately, Tutor commands generally react well to a variety of window sizes, and the developer can largely prevent a student from using an unacceptable window.

Figure 2 illustrates the difficulties that can befall a program permitted to run in a window that is too narrow.

THE CONVERSION PROCESS

The PC version of my software relies on a handful of subroutines. I began by converting the most essential three – all dealing with graph generation.

The logic of the routines carried over easily. The most difficult part – labelling axes – is essentially done for you in CMU Tutor by a set of specialized commands. In BASIC, I found it necessary to calculate the location of each number and tic mark, and then draw the number.

I was able to produce a labelled graph for a user entered function in an evening's work, shorter by many hours than the time required in BASIC. A portion of the time savings can certainly be attributed to having my PC version available, but some of the savings clearly follow from helpful diagnostics and ease in revision and testing.

My expectation is that the PC BASIC version of most instructional software can be readily converted to CMU Tutor. However, my own materials to date consist primarily of demonstrations requiring little interaction with the student. Rather than converting directly, the needed instructional interaction should be added, and I expect that to be much easier to accomplish in Tutor with the specialized commands noted above and the larger screen available.

FUTURE POSSIBILITIES

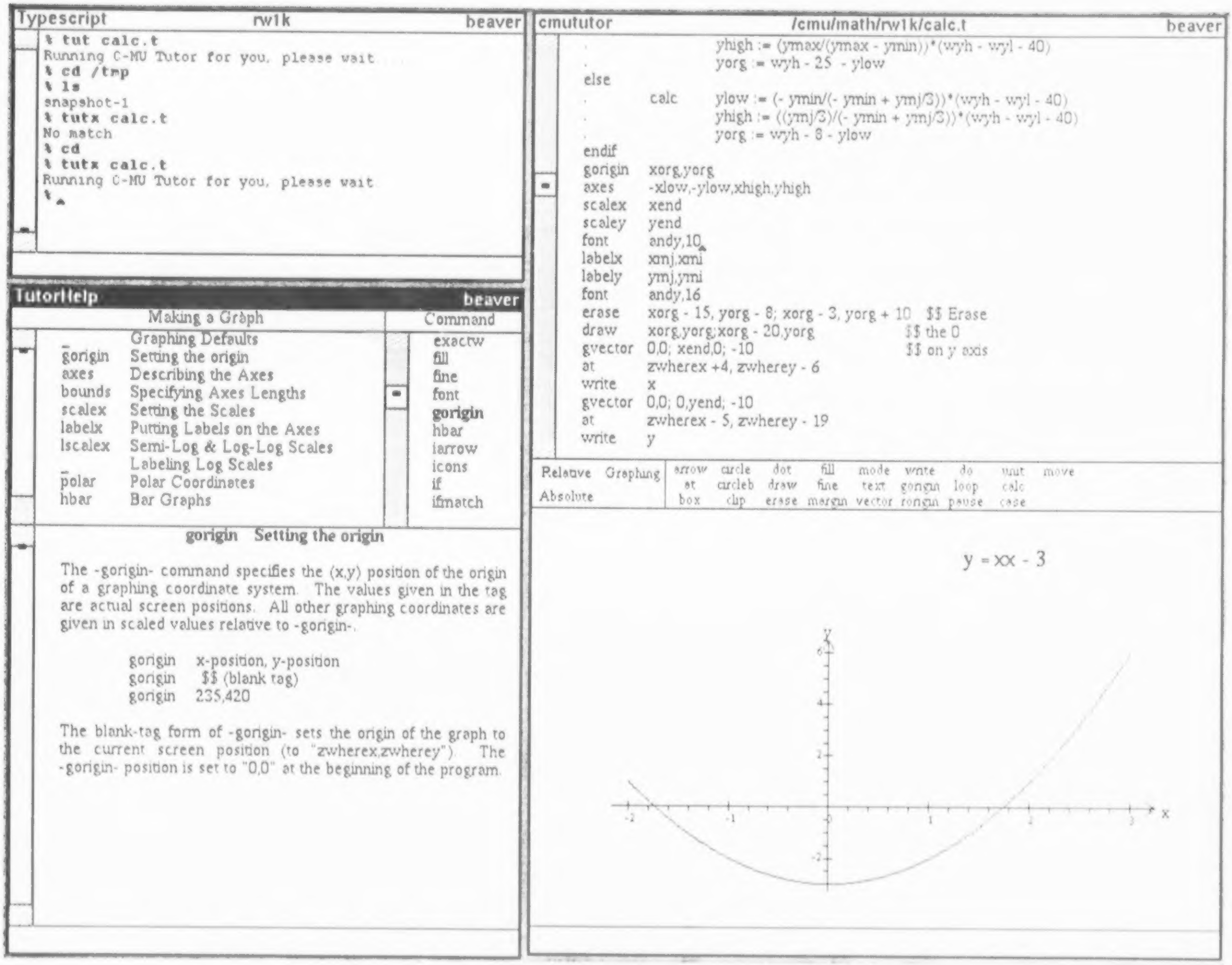
Neither ANDREW nor CMU Tutor is fully developed, but the expected features offer great promise for calculus software.

The ability to call outside routines from within a Tutor program will provide access to existing symbolic differentiation and integration programs. This will remove the limitation of the PC version to functions for which the derivatives are built into the program or for which the user enters the derivatives. Treatment of topics such as numerical integration, Taylor's formula, and root finding will be greatly enhanced.

Integration with the GRITS database on ANDREW will allow materials to be coordinated with a help text and a glossary.

Access to a variety of fonts will make the presentation of technical material easier as well as present greater visual variety.

The use of the mouse should allow much more rapid and flexible student response to options.



```
% tut calc.t
Running C-MU Tutor for you, please wait
% cd /tmp
% ls
snapshot-1
% tutx calc.t
No match
% cd
% tutx calc.t
Running C-MU Tutor for you, please wait
%
```

	Graphing Defaults
gorigin	Setting the origin
axes	Describing the Axes
bounds	Specifying Axes Lengths
scalex	Setting the Scales
labelx	Putting Labels on the Axes
lscalex	Semi-Log & Log-Log Scales
	Labeling Log Scales
polar	Polar Coordinates
hbar	Bar Graphs

gorigin Setting the origin

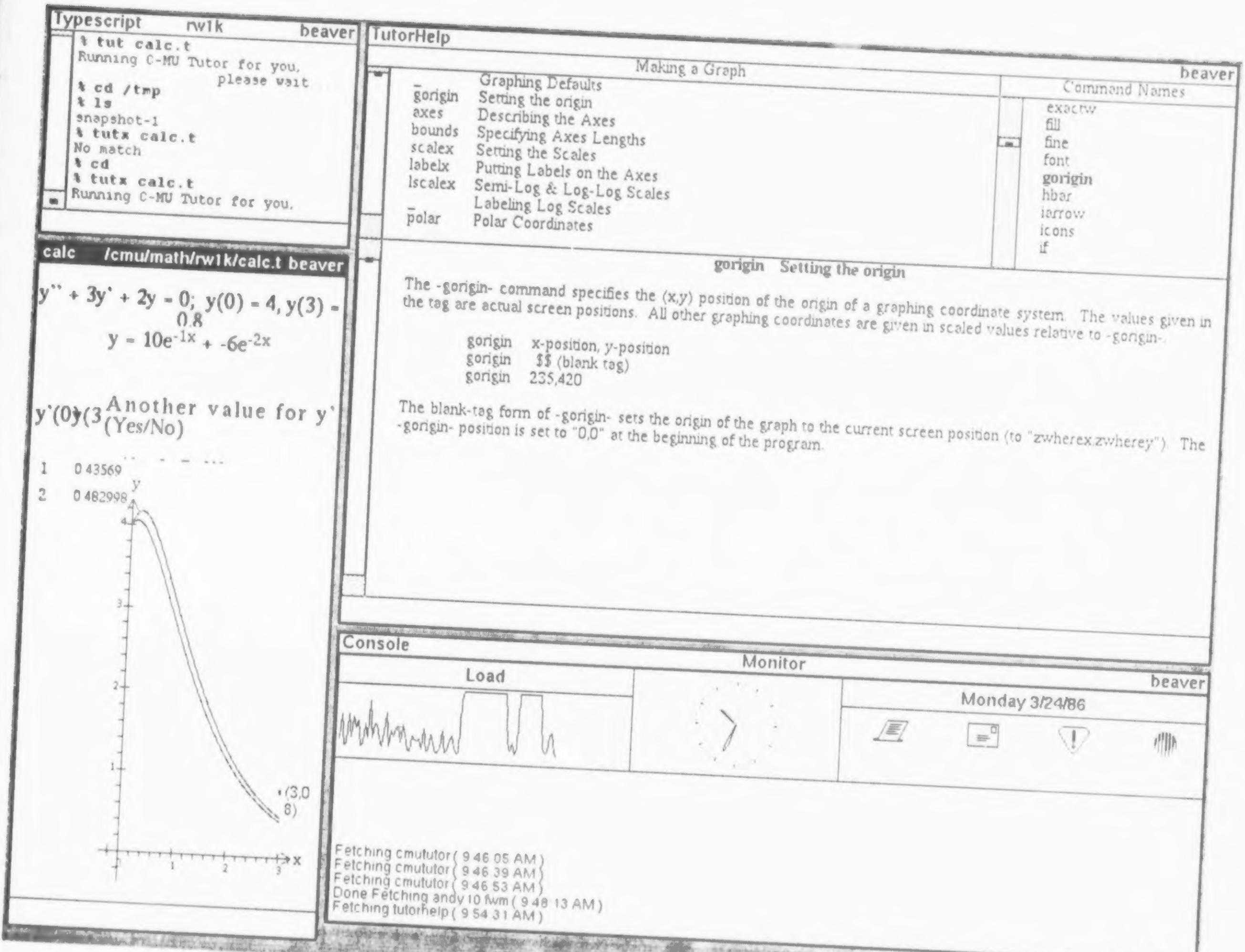
The -gorigin- command specifies the (x,y) position of the origin of a graphing coordinate system. The values given in the tag are actual screen positions. All other graphing coordinates are given in scaled values relative to -gorigin-.

```
gorigin   x-position, y-position
gorigin    $$ (blank tag)
gorigin   235,420
```

The blank-tag form of -gorigin- sets the origin of the graph to the current screen position (to "zwherex,zwherey"). The -gorigin- position is set to "0,0" at the beginning of the program.

```
.                      yhigh := (ymax/(ymax - ymin))*(wyh - wyl - 40)
.                      yorg := wyh - 25  - ylow
else
.            calc      ylow := (- ymin/(- ymin + ymj/3))*(wyh - wyl - 40)
.                      yhigh := ((ymj/3)/(- ymin + ymj/3))*(wyh - wyl - 40)
.                      yorg := wyh - 8 - ylow
endif
gorigin   xorg,yorg
axes      -xlow,-ylow,xhigh,yhigh
scalex    xend
scaley    yend
font      andy,10
labelx    xmj,xmi
labely    ymj,ymi
font      andy,16
erase     xorg - 15, yorg - 8; xorg - 3, yorg + 10   $$ Erase
draw      xorg,yorg;xorg - 20,yorg                   $$ the 0
gvector   0,0; xend,0; -10                           $$ on y axis
at        zwherex +4, zwherey - 6
write     x
gvector   0,0; 0,yend; -10
at        zwherex - 5, zwherey - 19
write     y
```

Figure 1.



```
% tut calc.t
Running C-MU Tutor for you,
                    please wait
% cd /tmp
% ls
snapshot-1
% tutx calc.t
No match
% cd
% tutx calc.t
Running C-MU Tutor for you.
```

$y'' + 3y' + 2y = 0;\ y(0) = 4, y(3) = 0.8$

$y = 10e^{-1x} + -6e^{-2x}$

$y'(0)$ $y(3$ Another value for y' (Yes/No)

1 0.43569

2 0.482998

	Graphing Defaults
gorigin	Setting the origin
axes	Describing the Axes
bounds	Specifying Axes Lengths
scalex	Setting the Scales
labelx	Putting Labels on the Axes
lscalex	Semi-Log & Log-Log Scales
	Labeling Log Scales
polar	Polar Coordinates

gorigin Setting the origin

The -gorigin- command specifies the (x,y) position of the origin of a graphing coordinate system. The values given in the tag are actual screen positions. All other graphing coordinates are given in scaled values relative to -gorigin-.

```
gorigin   x-position, y-position
gorigin    $$ (blank tag)
gorigin   235,420
```

The blank-tag form of -gorigin- sets the origin of the graph to the current screen position (to "zwherex,zwherey"). The -gorigin- position is set to "0,0" at the beginning of the program.

```
Fetching cmututor ( 9 46 05 AM )
Fetching cmututor ( 9 46 39 AM )
Fetching cmututor ( 9 46 53 AM )
Done Fetching andy10 fwm ( 9 48 13 AM )
Fetching tutorhelp ( 9 54 31 AM )
```

Figure 2.

COMPUTER AIDED INSTRUCTION IN NEUROANATOMY

Presenter: Steven E. Brauth *University of Maryland*

INTRODUCTION

Neuroanatomy is an usually difficult subject for most students to master. A major barrier for students attempting to master this field is the highly detailed and esoteric nature of the material. Another is the fact that students must be able to visualize the three dimensional appearance of brain structures in order to appreciate the anatomical relationships between them. The goal of this project is to develop computer–based tools for learning which will help students to improve their performance in courses in which neuroanatomy is an important part of the curriculum. Two types of computer–based tools have been developed thus far. The first consists of interactive tutorials written with IBM PILOT, an interpretive authoring language. These tutorials combine color graphics and text in a question and answer format and provide students with frequent feedback concerning the progress they are making. The second tool for learning is a software model of the human brain which combines interactive graphics with a large data base of information about the component structures of the human brain. Students can select from 30 views of the brain and move a cursor around the display space. The names and major anatomical features of brain areas can be quickly identified. In addition, the brain model program can sim– ulate the effects of brain damage and make predictions about the consequences of these "lesions". Students can test their knowledge of neuroanatomy by comparing their predictions to that of the computer.

These tools for learning have already been pilot tested in both graduate and undergraduate courses. The results indicate a high level of student satisfaction and interest. Furthermore, item analysis of exams indicates that the programs work, and can improve learning, even for the less talented students in the class.

I. USING IBM PILOT FOR COMPUTER AIDED INSTRUCTION

IBM PILOT is an authoring language designed for producing computer aided instruction (CAI) lessons combining graphics and text. I used PILOT to construct one series of lessons aimed at University of Maryland undergraduate students and another series of lessons aimed at University of Maryland graduate students. Undergraduates were enrolled in a course entitled "Biological Basis of Behavior". This course provides an overview of brain structure and function as well as a treatment of neurophysiology and neurochemistry. Graduate students were enrolled in a course entitled "Human Biopsychology". The goal of this graduate course is to provide psychology graduate students with a background in the neurosciences. Most of the graduate students enrolled in this course do so in order to meet departmental requirements for breadth in psychology. A large number are clinical and counseling psychology students who will intern in medical settings such as VA hospitals and other institutions with diverse patient populations. For this reason much of the material in Human Biopsychology touches on clinically relevant issues such as the effects of brain damage and mechanisms of neurological disease. In contrast, the course material for "Biological Basis of Behavior" emphasizes general issues such as the organization of sensory systems and brain biochemistry. The lessons developed for the two courses therefore reflect very different learning objec- tives.

B. TECHNICAL ISSUES – CONTENT OF CAI LESSONS

Both the undergraduate and graduate CAI PILOT lessons are menu driven. Students begin by selecting a topic of interest and following the directions that are provided. The lessons are entirely self–paced. Students are prompted to press particular keys in order to continue the lesson or answer questions. The topics for the "Biological Basis of Behavior" course are as follows: (1) Introduction, (2) The Human Brain, (3)Functions of the Brain, (4) Attention and Memory, (5) Evaluation. The last category (evaluation) provides feedback to the student. It is provided in the form of the proportion of correctly answered questions in three areas: general information, conceptual understanding and spatial relations. The students are advised to consult with their TA (teaching assistant) if performance in any one area is below 75%. For the graduate course ("Human Biopsychology") the topics include (1) Introduction, (2) Cranial nerves 3,4, 6 (3) Extraocular muscles, (4) Visceromotor control, (5) Next menu (6) Evaluation. The second menu for "Human Bio– psychology" includes the following topics: (1) Localization of Function, (2) Cortical Organization, (3) Sensory and Motor Cortex, (4) Neuropsychology Review and (5) Evaluation. Students can terminate a lesson at many points and return to the relevant menu in order to receive feedback or to choose another topic.

The format of the CAI lessons is that of an interactive tutorial. Information is presented in the form of short paragraphs and students are prompted to press any key in order to continue. After each group of paragraphs, questions are presented and students are prompted to either choose the correct answer in a multiple choice format or type in the correct answer from the keyboard. Feedback is provided, and students are frequently given the option of cycling back to material which had been presented previously, if answers are not correct.

The topics selected for the CAI modules are specific to the courses involved and were designed to reinforce the material presented in class and covered in the text. For example, the material for the "Biological Basis of Behavior" course provides many views of the brain. One of the teaching techniques used is to ask the student to name a structure which had been identified previously, but is now presented from a different three–dimensional perspective. Since this type of drill is not provided by the text and is virtually impossible in a classroom setting (both for technical reasons and because individual instruction is required), the CAI lessons provide genuine enhancement to the course.

C. RESULTS

The effectiveness of the CAI PILOT modules as teaching tools was assessed both by subjective "course evaluation" type questionnaires and by item analysis of examinations. The subjective judgments of the students concerning the PILOT modules was always positive. In November of 1985 students were asked to rate a series of descriptive statements concerning the CAI lessons on a five point scale (5=AGREE STRONGLY.....1=DISAGREE STRONGLY). Statements such as "On the whole I found the computer learning modules helpful" (average rating = 4.2115) and "I like the format of the computer modules" (average rating = 4.3462) were rated highest. Other statements in the survey were "The feedback provided was appropriate and helpful" (average rating = 4.0192), "I believe I learned more with the computer modules" (average rating = 4.077) and "The computers were accessible and easy to use (average rating = 4.077). Clearly the students found the CAI modules to their liking. This semester a similar questionnaire was distributed to the undergraduate students with similar results. The statement "The computer programs were extremely helpful in preparing for the first examination" received the highest rating (average rating = 4.59). Lowest rated were the statements "The computers were accessible at the times I was able to use them" (average rating = 3.67) and "The amount of material covered by the programs was satisfactory for my needs" (average rating = 3.66). The statements "The amount of detail and complexity of the computer lessons were appropriate for the course" (average rating = 4.26) and "The feedback provided by the programs concerning how well I learned the material was satisfactory for my needs" (average rating = 4.16) received reasonably high ratings from the students. Verbatim comments from the students included requests for more modules (particularly for physiology and neurochemistry) and requests for more opportunities to use the machines. We received more complaints this semester than last concerning access the the computers. This is probably due to the fact that all of our machines (seven ATs with enhanced graphics adapters) are set up in a single computer terminal room which was understaffed this semester. Security considerations dictate that students cannot use the computers unless there is a proctor present in the computer terminal room.

Item analyses of examinations bear out the contention that the computer lessons actually work. The first examination in the "Biological Basis of Behavior" course consisted of 50 multiple choice questions of which 13 were based on (but not duplicates of) the questions provided in the CAI modules. Overall class performance was much higher on the CAI-based questions than on the other questions (82% of CAI-based questions answered correctly versus 69% of other questions answered correctly). Of particular significance is the fact that both the upper and lower halves of the class (as defined by overall performance on the examination) profited from the CAI lessons. In fact the students comprising the lower end of the distribution actually showed slightly greater improvement! This means that these students (who are usually less motivated and less inclined to study effectively) can benefit from computer aided instruction. A summary of the results is presented below:

RESULTS OF ITEM ANALYSIS FOR PSYC 301 - SPRING 1986:

Top Half of Distribution (N=56 Students)

Question Type:	No. Quest.	No. Correct Responses	No. Wrong Responses	Total Responses
CAI-Related	13	677 (93%)	51	728
Non-CAI	37	1672 (80%)	400	2072

Bottom Half of Distribution: (N=56 Students)

CAI-Related	13	524 (72%)	204	728
Non CAI	37	1180 (57%)	892	2072

Combined (Both Halves of Distribution): (N=112 Students)

CAI-Related	13	1201 (82%)	255	1456
Non-CAI	37	2852 (69%)	1292	4144

Note that students in both the upper and lower half of the distribution scored about one letter grade better on the CAI- based questions than on the other questions. In the case of the brighter students this represents the difference between "A" level and "B" level performance. For the less gifted students this is the difference between a "D-" grade and a "C". The possibility that computer aided instruction can be used as a tool for remediation under certain circumstances can not be overlooked!

II. ENCEPHALOMODEL - A Software Model of the Human Brain

In addition to CAI modules based on IBM pilot, I am testing a program called "ENCEPHALOMODEL" this semester in the "Human Biopsychology" graduate course. This program is specifically geared to the needs of graduate and professional students seeking training in clinically relevent areas of neuroscience. The ENCEPHALOMODEL program is written in compiled BASIC (Microsoft Quick Basic compiler) with eight assembly language drivers to control the display. Students can select from 30 views of the human brain and request information about the structures which are visible. In addition the model program can simulate the effects of brain damage and make predictions about the clinical signs and symptoms which would result from these "lesions". The model program is interactive - students can test out their own predictions and compare them with that of the computer.

A. EDUCATIONAL PHILOSOPHY

Learning neuroanatomy involves forming a large number of new associations, particularly those between the verbal label attached to a particularly neural structure and the three dimensional appearance of the structure. Encephalomodel allows students to view the brain from different perspectives and develop associations between the name and visual apperance of parts of the brain. In addition, the lesion simulating programs incorporated into

ENCEPHALOMODEL allow the student to form other associations between the name of the structure, the visual appearance of the structure and the signs and symptoms commonly associated with focal neurological disorders.

The lesion simulating programs work as a kind of drill. One subprogram presents "symptoms" to the student and prompts the student to predict where the brain damage is most likely to be localized. This program fits in well with the CAI lessons on localization of function (see description of menus for PILOT lessons). When ready, the student presses any key. A drawing of the brain then appears. The symptom description (actually a slightly modified description incorporating technical terms) is repeated across the top of the screen. An arrow is positioned in the lower left hand part of the display field and the student is prompted to move the arrow with the cursor control keys to the general region probably involved in the computer– simulated pathology. When the ENTER key is pressed the computer provides feedback. If the answer is incorrect, the student is given up to two more tries to locate the brain area most likely to be responsible for the clinical pattern. After three tries (or after a correct response) the computer shades the entire structure and prompts the student "Do you know the name of this structure?". Students can then press any key to learn the name of the structure and verify their own assessments of the probable nature of the brain lesions underlying the computer simulated pattern of clinical signs and symptoms.

Another lesion simulating program in ENCEPHALOMODEL presents "lesions" in the form of crosshatching on a brain drawing and prompts the student to predict the consequences. The computer's assessments of the damage can be learned by pressing any key. The descriptions of the symptoms are provided in technical language (i.e. "homonymous hemianopia" refers to half–field blindness, "comprehensive aphasia" refers to loss of understanding of speech, etc.). Each symptom is prefaced by one of three descriptors: likely, probable or possible. Very heavy "damage" in a particular region results in the descriptor "likely" be applied to the symptom description. The descriptor "probable" implies at least 50% of the structure is damaged. Simulated damage involving about 25% of a structure results in the descriptor "possible" being applied. If no structure sustains significant "damage", the student is told that lesions are "small and scattered with little clinical effects".

In addition to providing the student with an opportunity to form verbal spatial associations, ENCEPHALOMODEL includes an ASCII file called BRAIN.DOC which is not part of the binary file (all of the symptom patterns, lesion signs and structure descriptors are incorporated into the .EXE file). BRAIN.DOC is made up of abstracts of from 30–60 words about each structure. Students can learn more about a structure by pressing the F10 function key. The image of the brain drawing disappears and is replaced by an abstract providing more information about the neural structure under the cursor. The advantage of making this ENCEPHALOPEDIA external to the main program is that it can be easily modified and adapted to different courses (for example a pharamcology emphasis for a course featuring material on brain chemistry).

ENCEPHALOMODEL also contains a help menu and a brief tutorial for students not experienced in its use. The program is menu driven. The main menu provides six options: Identify Discrete Structures, Associate Structure with Function, Simulate Lesions, Ask for Help, Set Up Display Options and Return to Operating System. The Identify Discrete Structures program allows students to choose views of the lateral surface of the brain and move a cursor around the display space. Pressing the ? key evokes the name of the structure under the cursor, the side of the brain involved and a brief 80 column description of the structure. Pressing the ENTER key shades the entire structure pointed to by the cursor. Pressing the SPACEBAR advances to the next image in the series. Pressing the \ key goes back to the previous image or moves one image back in the series. Students can shift to the top of the brain (t or T), the bottom of the brain (b or B), the left side view (l or L), the right side view (r or R), the first view in the series which is the occipital view (Home key) or the front end of the brain (End key). Pressing the ESC key returns to the main menu.

In order to improve the appearance of the display and to make the display as visually interesting as possible, ENCEPHALO- MODEL gives the user control over the colors. Pressing the c key changes the color of the background. Pressing F8 shades the brain image, making it appear more like a solid object. Repeated keypresses of c or F8 change the colors so that a large number of possible combinations can be achieved (16X16=256). Pressing the shifted C resets all the color parameters. The background is then dark and the brain image drawn in yellow lines with no internal shading. Text is bright white. Pressing F9 brings up the help menu. Pressing F10 brings up an abstract from the ENCEPHALOPEDIA. Pressing F1 evokes the lesion simulating program. Pressing F2 evokes the associate structure and function program (i.e. computer presents symptoms and allows the student to find the structures probably involved).

III. TECHNICAL ISSUES – PROGRAMMING TECHNIQUES

IBM PILOT is an interpretive language designed to work with the color graphics adapter and IBM PC with at least 128K of memory. In order to work with the enhanced graphics adapter (EGA) it was necessary to provide assembly language drivers to display images in the higher resolution color graphics modes possible with the EGA. These programs are assembled into object files with the MASM assembler, linked to .EXE files with the LINK program and converted to binary files (.BIN) using the EXE2BIN utility program. The resulting binary files are treated as string variables by pilot and executed using the V: command (see PILOT manual, pages A–39 and A–40). An excellent example of the successful application of these techniques is the SQZ utility (SQZ.BIN) developed by Larry Kheriaty as a quick draw program for PILOT graphics. Images are "squeezed" into a compressed format by coding for the redundancy in the image. The "squeezed" image is then represented as a binary file with the SQZ.BIN utility as a header. PILOT reads the entire image file as a string variable and executes the header portion. The compressed image is treated as data by the SQZ.BIN procedure which then decompresses the image into the screen buffer at segment paragraph address hexadecimal B800 (B800h). In order to display images in the high resolution EGA graphics modes (specifically modes 14 and 16), I wrote two .BIN utilities. The first is called SETMODE.BIN, and the other is called TRANS.BIN.

SETMODE.BIN is a simple program which sets the desired BIOS video mode. The default mode is mode 14, a 640X200 pixel all– points addressable graphics mode compatible with the enhanced graphics adapter. The mode can be changed by altering the second byte in the setmode.bin string. PILOT does this using the C: command (see PILOT manual pages A–13 and A–14). The TRANS.BIN utility is more complicated. For the BIOS graphic video modes compatible with the color graphics card (modes 1–6), odd and even scan lines for graphics images are stored in separate banks of memory. For the EGA graphics modes (modes 13–16), odd and even scan lines alternate (as one would logically expect!). Thus images prepared for display with the CGA cannot be displayed in the higher resolution EGA modes unless the odd and even scan lines are "unscrambled". TRANS.BIN does this and allows an image prepared for the CGA to be transferred to modes 14 or 16 and displayed with higher resolution and more colors. To do this, TRANS.BIN sends data directly to the EGA screen buffer at segment paragraph address hexadecimal A000h (A000h) and uses the EGA graphics controller register in order to set the bitmask and map mask registers (T. Hoffman, PC Tech J., Dec. 1984,pp.58–77;R. Wilton, BYTE, Fall 1985 pp. 209–220). TRANS.BIN transfers the data in the image to the EGA one byte at a time and sets the color. A complex image can be prepared in the form of "overlays", each consisting of pixels with the same color value.

TECHNICAL CONSIDERATIONS - ENCEPHALOMODEL

ENCEPHALOMODEL uses seven drivers to produce the display plus the SETMODE procedure also used with the PILOT modules. For compiled basic an .OBJ file is produced by the compiler. This program passes parameters to subroutines by pushing addresses onto the stack. The final .EXE file is produced by the LINK.EXE program. The other assembly language procedures are called LOADS, TRANSALL, SCANR, FILLS, CURSR, ARROW and DOTS2. LOADS uses DOS function calls to load image files and BRAIN.DOC into memory. TRANSALL reads a compressed version of the brain image (not in the same format as the SQZ utility of PILOT) and sends data to either the EGA screen buffer and registers or to the CGA screen buffer depending upon the display mode. Thus students owning IBM PCs with the Color Graphics Adapter could make use of the ENCEPHALOMODEL program, although the displays produced with the CGA are less colorful and interesting than those produced with the EGA. The main menu option called "Set up Display Options" allows students to specify the display adapter and monitor they are using. Encephalomodel then sets the appropriate BIOS video mode automatically. SCANR searches the image data file to find all pixels corresponding to a particular structure. This allows structures to be shaded in their entirety. SCANR xors the shading into the image so that the shading can be removed if the student desires. CURSR puts a cursor in the form of a magenta cross on the screen. ARROW puts a white arrow on the screen and DOTS2 provides the crosshatching for the lesion simulating program. All display drivers sense the BIOS video mode and can work in CGA compatible as well as EGA specific modes. The BIOS modes supported by ENCEPHALOMODEL are modes 4,6,14 and 16.

IV. OVERALL EVALUATION

Computer aided instruction has already proved to be an effective tool for learning at the University of Maryland. It remains to be seen if CAI can work as a remedial aid and more program development is needed. Eventually all aspects of the undergraduate "Biological Basis of Behavior" course will be represented by computer learning modules. As the ENCEPHALO- MODEL program is improved it will serve even more advanced applications. For example, plans call for expanding the program so that it will be able to stack brain sections. This will enable the program to represent the internal as well as the surface features of the brain and will provide a unique tool for helping students to visualize the three dimensional anatony of brain circuits.

The computer can also be an aid to the instructor as well. I taught the "Biological Basis of Behavior" course this semester. It was the first time I had done so in about two years and it was the first time I had ever taught an undergraduate course with computer based instruction. From the instructor's point of view CAI has a marvelous appeal. For the first time in my teaching career (ten and half years) I had an answer to the question asked most frequently by undergraduate students of biopsychology– how much of this is really important?'. Students are easily overwhelmed by the detail in even the least sophisticated textbooks. Authors and publishers often feel obliged to include as much detail as possible in order to impress potential buyers. Students are faced with complex diagrams of the autonomic nervous system, elaborate tables of the components of the cranial nerves and usually too few pictures depicting the brain in different orientations. When students asked the predictable question this semester, I had a good answer for them, at last. I suggested that they concentrate on the material covered by the computer programs. I wrote the programs, therefore the material emphasizes exactly what I want the students to learn. From the instructors point of view CAI is like having a "clone" of oneself available to provide some private tutoring. This not only provides an enhancement to the course but makes the job of undergraduate teaching a little bit easier as well!

BIBLIOGRAPHY:

1. Hoffman, Thomas V. "Graphic enhancement" in PC Tech Journal, April, 1985, volume 3, No. 4 pp58–77.

2. Wilton, Richard "Programming the Enhanced Graphics Adapter", BYTE Magazine, Fall 1985, pp. 209–220.

3. Kheriaty, L., "Using IBM Pilot", IBM Corporation, 1st Ed., September, 1984.

ACKNOWLEDGEMENTS:

The author would like to thank Drs. McDaniel, MacLeod and Weissman, University of Maryland Instructional Computing Programs for their advice and encouragement. Special thanks is due to Mr. Ben Chomiak for his artistic assistance in producing the drawings of the brain for the encephalomodel program.

Loss of precision and independent limb movement on left side of body
Do you know the name of this structure?

Right
Answer!!!

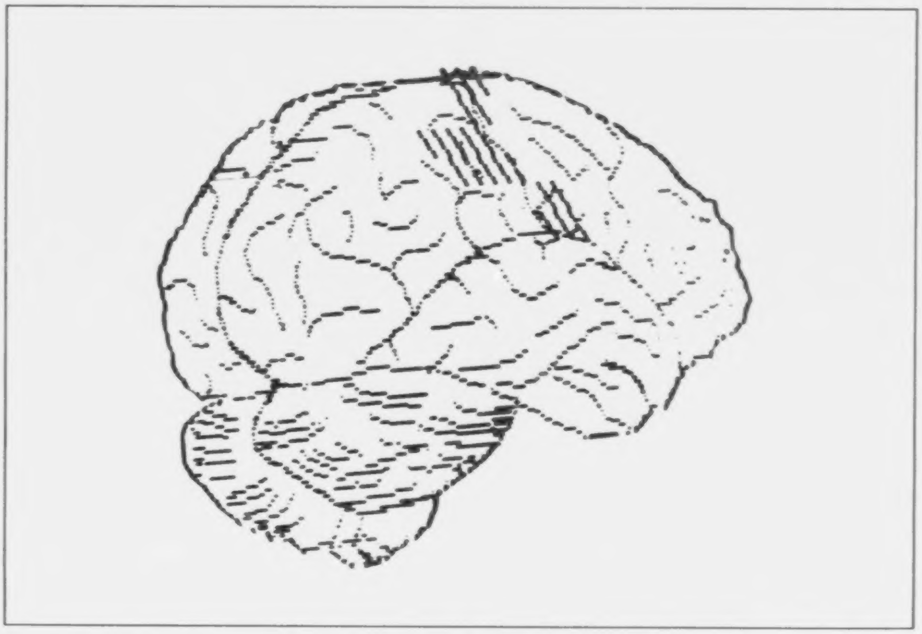


Figure 1.

PRESS ANY KEY TO FIND OUT

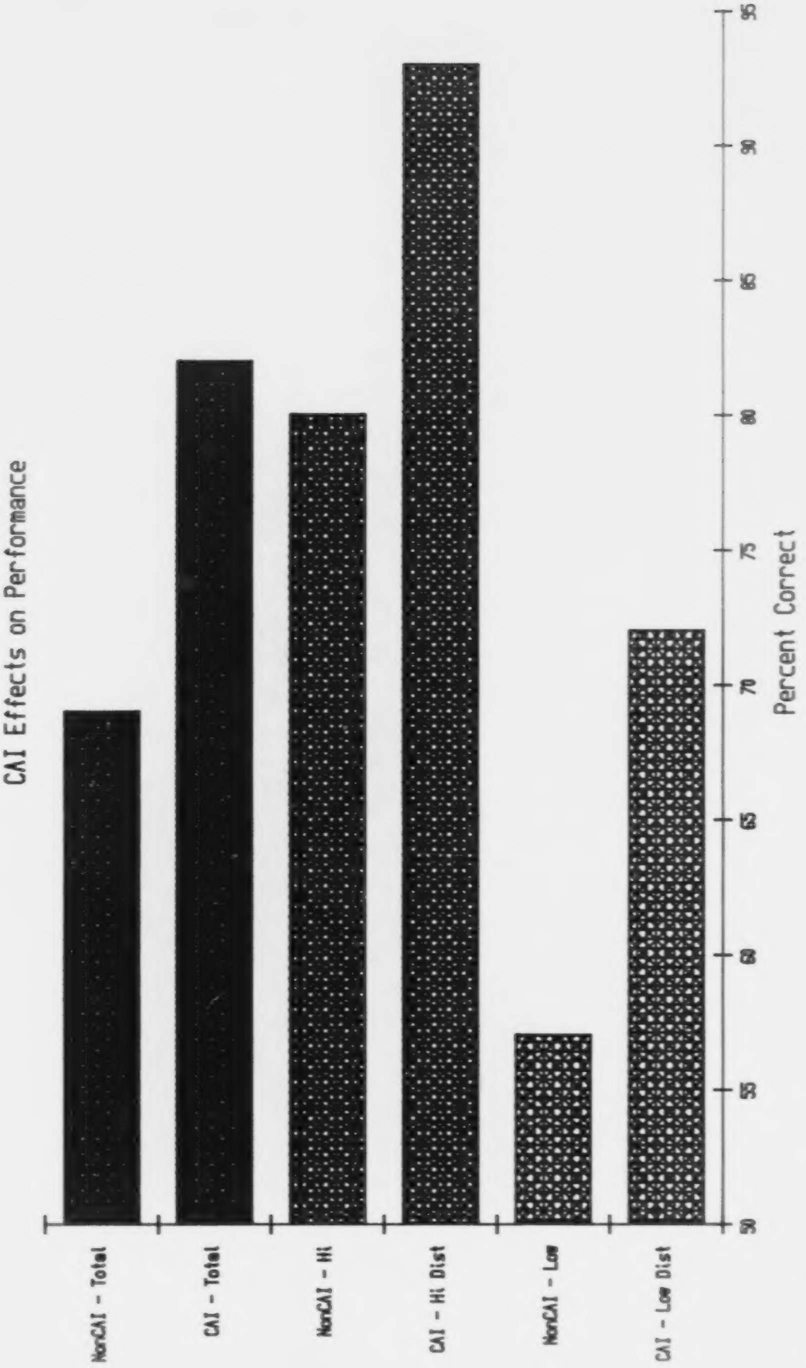

CAI Effects on Performance
Percent Correct
50
55
60
65
70
75
80
85
90
95
NonCAI - Total
CAI - Total
NonCAI - Hi
CAI - Hi Dist
NonCAI - Low
CAI - Low Dist

Figure 2.

COMPUTER GRAPHICS FOR AN INTERMEDIATE-LEVEL OPTICS COURSE

Presenter: Ned S. VanderVen *Carnegie-Mellon University*

INTRODUCTION

This paper describes three applications, covering selected topics from both geometrical and physical optics, intended for an intermediate level course. All of these programs make extensive use of the window- manager and base-editor facilities of the Andrew System developed in the Information Technology Center at Carnegie-Mellon University as a joint project with IBM.

COURSE DESCRIPTION

The course for which these applications were designed is a fall- semester elective, whose typical enrollment of twelve to fifteen students consists primarily of junior and senior Physics majors but usually includes two or three students from other science or engineering departments. There are three one-hour lecture-recitations and one three-hour laboratory period each week. The usual text is Optics, by Hecht and Zajac, which we follow selectively rather than sequentially, although we treat most of its material. Approximately three weeks at the beginning of the semester are devoted to geometrical optics; the remainder of the term is spent on wave optics. The students are assumed to have been exposed to Fourier methods, but the amount of exposure varies considerably. For our Physics majors, at least, this may be changing, because of the recent introduction of a Mathematical Physics course that will be elected by most of them in the fall of their junior year, and which will have covered Fourier methods by the time they appear in the optics course.

The treatment of geometrical optics begins with Gaussian matrix theory, and proceeds to a discussion of aberrations and numerical ray- tracing. The first application I shall describe is a flexible program that allows the user to carry out exact numerical ray-tracing calculations for a system consisting of an arbitrary number of spherical refracting surfaces, with graphical output of the results. The other two applications deal with Fraunhofer diffraction and Fourier methods.

THE ROLE OF COMPUTING IN THE COURSE

Until recently there had been little computing in this course, apart from a minimal amount of numerical analysis of laboratory data. I have tried to introduce some computing that will enable us to study topics that we would not otherwise undertake and to supplement material that is traditionally part of the course. The students are given a small number of programming assignments, primarily during the discussion of geometrical optics. One is to write a program to calculate the paraxial system matrix and the Gaussian constants for a system of several refracting surfaces. Another is to write a program for exact numerical ray–tracing through a single spherical refracting surface and to use it to calculate the spherical aberrations; this I consider sufficient to show them how numerical methods are applied in geometrical optics. They are also given assignments in which they must use a general ray–tracing program that I have written, originally for our main–frame time sharing system and more recently in Turbo Pascal for the IBM PC. The output of this program had been restricted to numerical tables, and it was the desire to add graphics that led to OWL, the Optics Workbench Laboratory, and the first application that I shall describe.

OWL, THE OPTICS WORKBENCH LABORATORY

This program, written in C, was named by Eric Shulman, the student who did a major part of the programming. With it the user may construct a refracting optical system out of a small number of basic elements, display it in a cross–sectional view, carry out exact numerical ray– tracing calculations for it, and display the trajectories of the rays. The parameters of the system may be rapidly and conveniently altered, the ray–tracing repeated, and any changes observed. OWL runs in a single Andrew window divided into three regions. Of these, the two smaller contain scrollable text that can be edited with a general– purpose editor, while the third and largest is a graphics region. One text region is used for a lesson document containing both general instructions and specific applications; the other contains the "setup document" describing the optical system being studied. Example setup documents can be cut from the lesson document, or from additional lesson files in other windows, and pasted into the setup region with the screen editor.

An optical system is constructed by placing elements along an axis (the optical bench). With OWL, the user constructs a system by entering a description of the elements into the setup region. The standard element types are a simple lens, a cemented doublet, and a planar aperture stop, which may be combined in any way to construct a complex multi–element system. All refracting surfaces must be spherical. Two additional elements are a planar "film", and a "lightbar" that serves as a source of incident parallel rays. Only one of each of these may be specified, and the film must also be the rightmost element of the system. The setup document for a Tessar, a three–element camera lens of modest quality, including an appropriate lightbar, is shown below:

```
lightbar at 0: height=10, angle=0, nrays=12

lens at 10: h=9, w=3.57, rL=16.28, rR=-275.7, i=1.6116

lens at 15.46: h=8, w=0.81, rL=-34.57, rR=15.82, i=1.6053

doublet at 19.52: h=8, wL=2.17, wR=3.96, rL=9999, rM=19.2, rR=-24, iL=1.5123, iR=1.6116
```

OWL recognizes three syntactical forms, varying in brevity, for each element, and these may be mixed in any setup document. In the example above, the intermediate form is used for the two lenses and the doublet, while the light bar is specified in long form. Although a film element may be initially specified in a setup document, it is not necessary to do so; if it is not, OWL will automatically place a film at the paraxial back focal plane in a manner that will be described below.

Having entered a setup document, the user may view a scaled drawing of the system by selecting "Construct Bench" from the menu in the graphics region. When this selection is made, a parser, written with the Unix utilities "lex" and "yacc", reads and compiles the setup document, translating the text into data and C code for drawing a picture of the system and for the numerical calculations required for ray–tracing. The data structures for these two processes differ because it is convenient to draw the system element–by–element, but ray–tracing should be carried out surface–by–surface. By successively editing the setup document and selecting "Construct Bench" from the menu, the user may alter the system design and view the changes in the drawing. At any time the optical performance of the system may be observed by activating the light bar to trace rays through it. In its present form OWL traces only meridional rays, but because the numerical ray–tracing is carried out with general skew–ray algorithms, the extension to skew rays will not be difficult, and we plan to add this feature before the program is used in the fall.

OWL offers considerable flexibility in placing the film element. If no film has been specified in the setup document, the selection of "Construct Bench" draws the system to a larger scale, because the horizontal dimension extends only to the last vertex. If no film has been specified when rays are first traced through the system, OWL calculates the location of the paraxial back focal point and places a film there, writing the film description into the setup document. It also rescales and redraws the system, including the film plane, traces rays through the system up to the film, and displays them on the drawing. If a film element is already specified when the lightbar is activated, OWL traces rays up to that film element, although it may no longer be located at the focal plane if other elements of the system have been changed. If no film has been specified and the system is divergent, OWL places the film at a reasonable distance beyond the last vertex.

Figure 1 is the print of a screen dump of OWL output for a Cooke triplet, a three–element camera lens similar to the Tessar. The lesson and setup regions are at the top. The drawing in the graphics region shows, to scale, both the triplet and the rays that have been traced through it up to the paraxial focal plane for a parallel bundle of rays incident parallel to the axis. Some of the various menu choices in the graphics region are also shown.

With OWL the students will have a flexible graphics tool with which to study the exact behavior of systems made up of elements having spherical refracting surfaces. They can study both simple lenses and multi–element systems, observing various aberrations for uncorrected systems, and some of the techniques by which these aberrations can be corrected. Among the problems that will be assigned are the following:

Calculation of the spherical aberrations of a simple converging lens of specified focal length, and the minimization of these aberrations by "bending".

The design of a cemented doublet, corrected so as to reduce the spherical aberration to zero for rays incident at a specified distance from the axis; calculation of the spherical aberration for rays at intermediate distances.

Observation of the further reduction of spherical aberrations by more highly corrected systems.

Other aberrations, such as coma and field curvature can also be observed. Although a thorough study of coma must await the addition of full skew–ray capability, some of its features may be observed by studying the behavior of meridional rays.

It can be seen from Fig. 1 for the Cooke triplet that even the high resolution of the Andrew workstation screen is not adequate to show the small residual spherical aberrations for a well–corrected system; to study these the students will have to supplement OWL with programs giving numerical output.

MAKING SINGLE SLITS WITH VARIABLE TRANSMISSION

In most introductory Physics courses the treatment of Fraunhofer diffraction by a single slit relies upon a simple geometrical method for deriving the spatial variation of the diffracted field and the resulting intensity. At the intermediate level, in which the treatment of diffraction is extended to a variety of apertures, more general analytical methods are developed which can be readily extended to situations in which the transmission varies across the aperture. If the spatial dependence of the transmission is written as a function of position variables ranging over the aperture, the amplitude of the diffracted field in the Fraunhofer limit is the spatial Fourier transform of this "aperture function". For diffraction by a single slit, the subsidiary maxima are seen to be the result of high– frequency spatial Fourier components in the diffracted field introduced by the abrupt discontinuity in the transmission at the edge of the slit. The application of Fourier methods to the treatment of Fraunhofer diffraction should be an important component of any modern intermediate–level optics course.

A particularly interesting application of Fourier optics is in the phenomenon of "apodization." If the aperture function is chosen so as to reduce the discontinuity in transmission at the edge of the slit, the fraction of high–frequency spatial Fourier components in the diffracted field is also reduced, with a corresponding reduction in the relative intensity of the subsidiary maxima. Some textbook examples of symmetric aperture functions having this property are a gaussian, one–half period of a cosine function, and the triangular function, in which the transmission falls linearly from a maximum at the center of the slit to zero at each edge. If any one of these is worked through, either in the textbook or in a lecture, others may be assigned as homework problems.

I have for some time wanted to be able to demonstrate apodization experimentally, or, better still, have the students carry out laboratory measurements on single slits with variable transmission, but have been unable to find any source of such slits. The problem of making slits of width 0.5 mm or less with precisely varying transmission is not trivial. We have approached it by using the powerful graphics facilities of the Andrew system to generate large (approximately 1.2 cm wide) slits in which the varying transmission is achieved by filling the slits with a suitably varied grey scale of randomly placed dots (each dot is one pixel). Beyond the edges of the slits all dots are filled in, so that the screen, which displays black on white, is entirely black. A raster file of the resulting screen display is generated, printed on a high–resolution Xerox laser printer, and the final slit made by photographing the print with high–speed Polaroid slide film at a reduction of approximately 40 to 1.

A program to generate these slits has been written by Judith Sherwood in the "cmututor" language. For each line of pixels the appropriate filling probability is computed for the chosen aperture function. For each pixel in the slit the program then generates a random number between 0 and 1 and the pixel is filled if that number is less than or equal to the filling probability. A typical slit 55 pixels wide by about 1000 pixels long produces a final slit, after photographic reduction, about 0.3 mm by 7 mm.

We have made several symmetric slits with triangular, gaussian, and staircase aperture functions. The staircase function, if the number of steps is large, is a good approximation to the triangular function. An actual print of a slit with a three–step staircase aperture function is shown in Fig. 2, and Fig. 3 shows its calculated Fraunhofer diffraction pattern. Even our "triangular" function is a twenty–seven step staircase with each step one pixel in width, as is our gaussian slit, which is also truncated.

Slits of variable transmission prepared by this method show, when illuminated by the beam from a helium–neon laser, a diffraction pattern with subsidiary maxima relatively weaker and a central peak broader than the diffraction pattern from slit with uniform transmission. In most cases it is possible, even without quantitative photometric scanning of the diffracted intensity, to distinguish between the different apodizing functions. The four–step function, however, is already a sufficiently good approximation to the triangular function that we will need a better microphotometer than we now have in order to distinguish between them.

PLOTTING OF FRAUNHOFER DIFFRACITON PATTERNS FROM COMPLICATED APERTURES

The graph of the Fraunhofer diffraction pattern in Fig. 3 was prepared with the C program "graph" written by Bruce Sherwood for the Andrew system. "Graph" provides three services: a plotter for graphing a function, a flexible desk calculator, and a program that can numerically integrate and plot the results for up to three simultaneous ordinary differential equations. The differential equation solver was added to "graph" by R. T. Schumacher. "graph" is available to the students in intermediate optics for the investigation of Fraunhofer diffraction patterns. With it, they can investigate the ordinary single, double, and multiple-slit diffraction patterns, as well as those from slits with variable transmission. The flexibility of graph allows the students to investigate a variety of multiple slit patterns obtained by varying the number of slits, their widths, and their spacings. In many of these examples, the diffraction pattern can be expressed as the product of a single slit diffraction factor and an interference factor of reasonably simple form. Confronted with problems of this sort, the students should be able to obtain many of the features of the diffraction pattern from free-hand sketches, and the cultivation of that facility is an important part of any course in physical optics. "graph" is a useful, but not essential, tool for these examples.

"graph" allows us to investigate with equal ease the diffraction for more complicated aperture functions, for example some of those mentioned in the preceding section, whose Fourier transforms can be calculated exactly but lead to expressions for the diffracted intensity sufficiently complicated that it is often difficult for students to visualize the graph of the resulting pattern. It is in these examples that a powerful graphing utility is most advantageous.

The staircase aperture functions provide an interesting set of illustrative examples. Their diffraction patterns can be calculated exactly; in fact they can be calculated in a variety of ways, each exhibiting some interesting aspect of the problem, and each leading to a different but equivalent expression. A slit of unit width and unit height having a three-step staircase aperture function can be viewed as the superposition of three single slits, each of height 1/3, and of widths 1, 0.6, and 0.2, but it can also be viewed as the superposition of a single slit and two double slits, or as the convolution of a single slit of width 0.2 with a row of delta-functions located at +/- 0.4, +/- 0.2, and 0 with heights, respectively, of 1/3, 2/3, and 1.

For any of these three approaches, calculation of the Fourier transform and an expression for the Fraunhofer diffraction pattern is straightforward, but the result is an expression sufficiently complicated that it is hard for a student to visualize the shape of the diffracted intensity. Moreover, the three expressions are so different in form that it becomes a tedious and unenlightening trigonometric exercise to demonstrate their equivalence. But it is easy to do so by plotting them with "graph." Two sets of expressions may be successively pasted into the program region of "graph" and the results from each superimposed on the same axes for comparison. The results for a uniform slit and an apodized slit may be compared in the same way.

While working on this material, I was led to look at the standard example of the double slit in what was, for me at least, a new way. Traditionally the double slit is viewed as the superposition of two slits of fixed width and of variable separation. But if apodization is the suppression of single–slit subsidiary maxima through reduction of the discontinuities at the edges, we may also view the double slit as a single slit into which an additional discontinuity has been deliberately introduced by the placement of a narrow mask at the center. The result for a very narrow mask, which can readily be studied with "graph", is an increase in the relative size of the subsidiary maxima, which continue to grow as the mask is increased in width.

I have separated these two aspects of Fraunhofer diffraction because they involve two different application programs operating in the Andrew environment, but this separation is artificial, and I intend to combine them for the students in a single extended assignment of the following steps:

Choose your own aperture function, determine its Fourier transform and the resulting Fraunhofer diffraction pattern, and plot out the result with "graph."

Using "cmututor", generate a slit with this aperture function and make a print of the screen dump.

Take a photograph of the print, suitably reduced, to make a diffracting slit, observe and measure its diffraction pattern, and compare the result with the calculated expression.

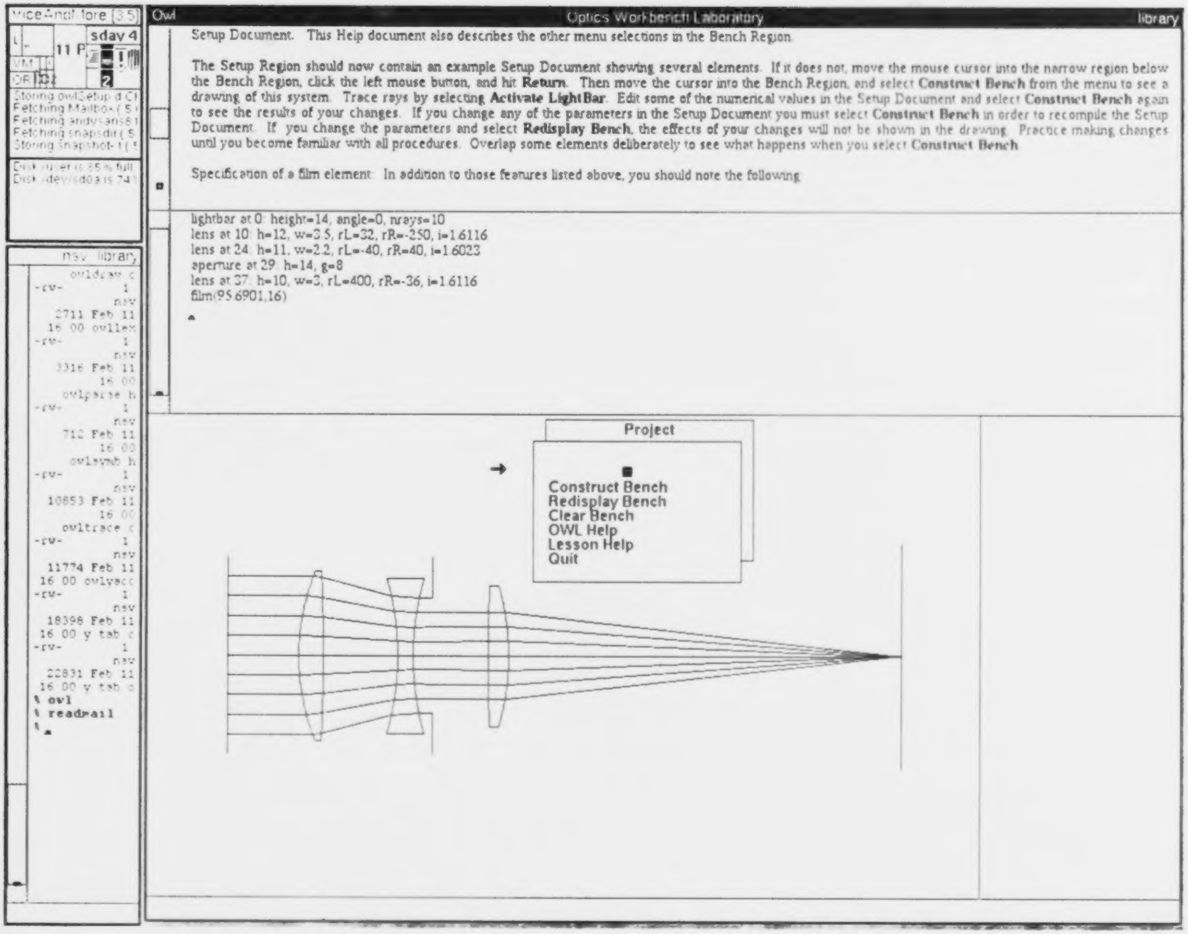
In this way the students will have the opportunity to apply a broad array of techniques to the solution of a problem that they have also helped to define.

CONCLUDING REMARKS

Taken altogether, these programs will be used during approximately three weeks of a fifteen–week course. In combination with the purely numerical ray–tracing programs, OWL will be used extensively for only about one week, perhaps a little longer, not because of inherent limitations, but because there is so much other material that must be covered. The applications to Fraunhofer diffraction are intended to supplement and enhance the usual treatment of the subject. Although computing is not likely to become a dominant feature of the course, it does provides us, for certain subject areas, tools to study topics that we would not otherwise cover, or to study standard topics in new and different ways.

ACKNOWLEDGEMENTS

Many people participated in this work, and I should like to acknowledge their contributions. Eric Shulman coined the name and did much of the programming for OWL, and made significant contributions to its overall design. Judith Sherwood, who has been active in the development of the "cmututor" language, also wrote all the "cmututor" programs for generating the slits with variable transmission. Barry Luokkala did all the photographic work. Chris Koenigsberg was responsible for porting several programs to the IBM RT from the Sun workstations on which they were originally developed. Bruce Sherwood's contributions have been manifold; "graph" and "cmututor" are only the most apparent of his efforts to make the Andrew system more accessible to those of us using it for the development of educational software.



```
Storing owlSetup d Ch
Fetching Mailbox ( 5
Fetching andysans8 1
Fetching snapsdir ( 5
Storing snapshot-1 ( 5
```

```
Disk /user is 85% full
Disk /dev/sd0a is 74
```

Setup Document. This Help document also describes the other menu selections in the Bench Region.

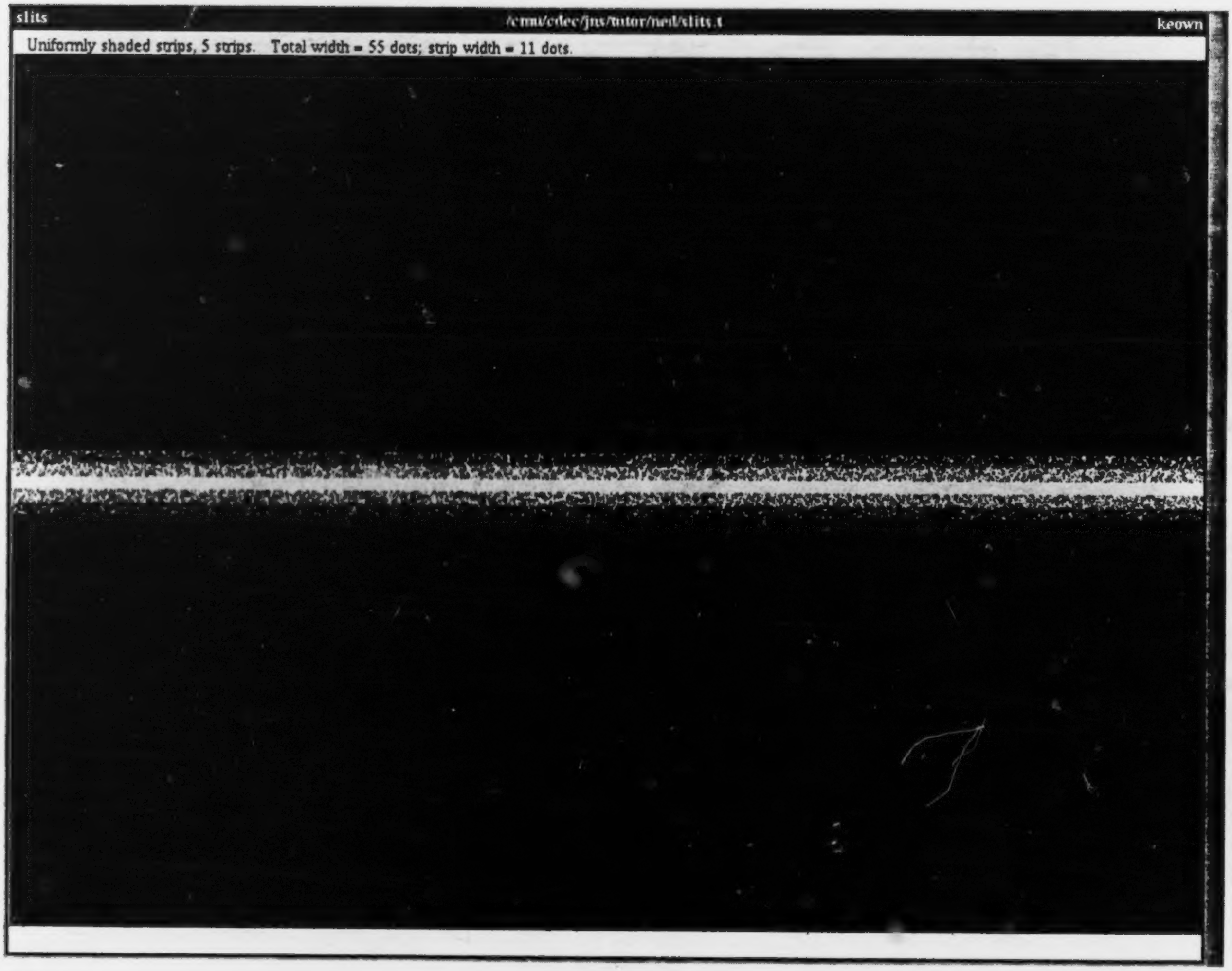
The Setup Region should now contain an example Setup Document showing several elements. If it does not, move the mouse cursor into the narrow region below the Bench Region, click the left mouse button, and hit **Return**. Then move the cursor into the Bench Region, and select **Construct Bench** from the menu to see a drawing of this system. Trace rays by selecting **Activate LightBar**. Edit some of the numerical values in the Setup Document and select **Construct Bench** again to see the results of your changes. If you change any of the parameters in the Setup Document you must select **Construct Bench** in order to recompile the Setup Document. If you change the parameters and select **Redisplay Bench**, the effects of your changes will not be shown in the drawing. Practice making changes until you become familiar with all procedures. Overlap some elements deliberately to see what happens when you select **Construct Bench**.

Specification of a film element: In addition to those features listed above, you should note the following:

```
lightbar at 0: height=14, angle=0, nrays=10
lens at 10: h=12, w=3.5, rL=32, rR=-250, i=1.6116
lens at 24: h=11, w=2.2, rL=-40, rR=40, i=1.6023
aperture at 29: h=14, g=8
lens at 37: h=10, w=3, rL=400, rR=-36, i=1.6116
film(95.6901,16)
```

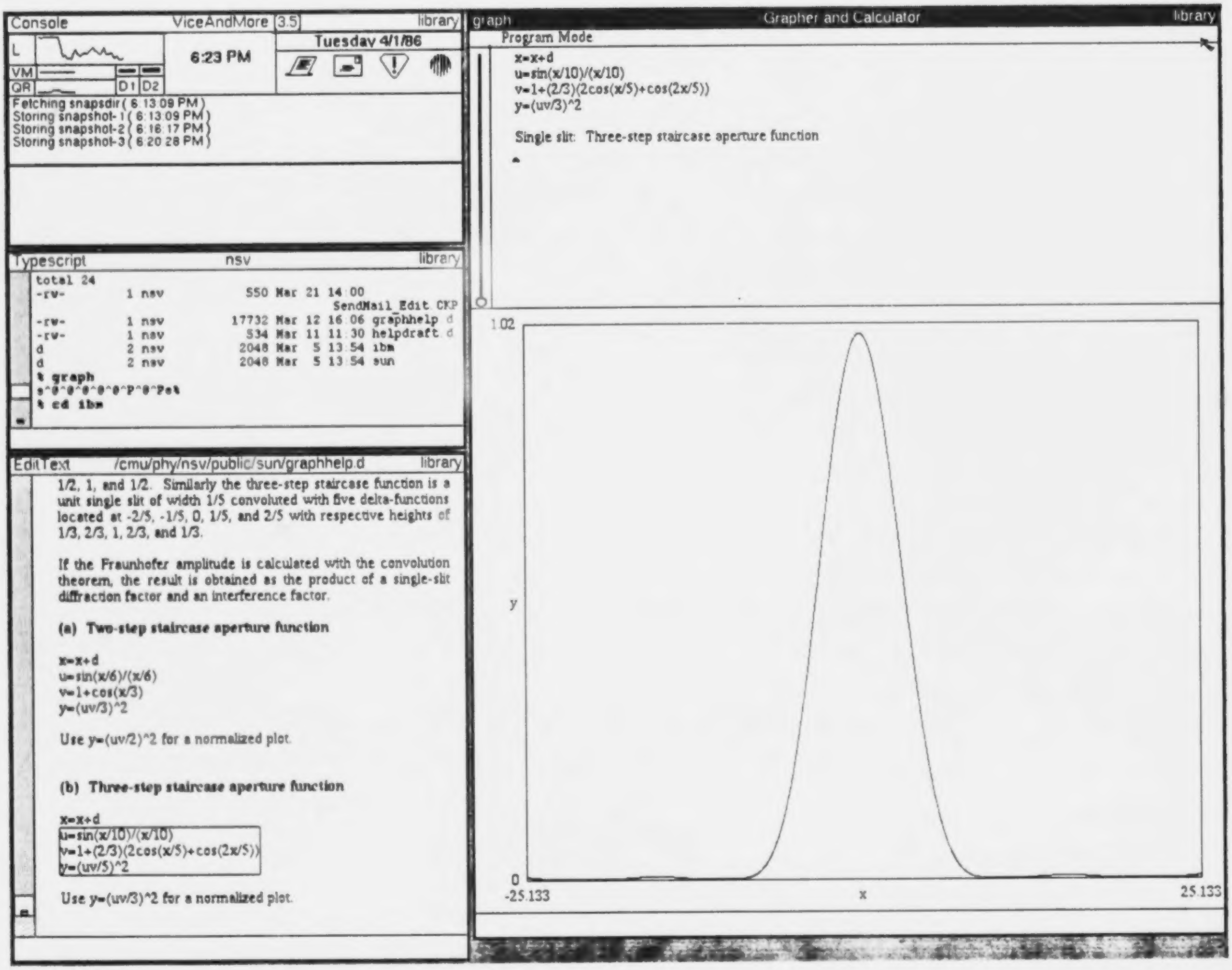
```
      owldraw c
-rw-          1
            nsv
   2711 Feb 11
 16 00 owllex
-rw-          1
            nsv
   3316 Feb 11
          16 00
    owlparse h
-rw-          1
            nsv
    712 Feb 11
          16 00
     owlsymb h
-rw-          1
            nsv
  10853 Feb 11
          16 00
    owltrace c
-rw-          1
            nsv
  11774 Feb 11
 16 00 owlyacc
-rw-          1
            nsv
  18398 Feb 11
 16 00 y tab c
-rw-          1
            nsv
  22831 Feb 11
 16 00 y tab o
% owl
% readmail
%
```

Figure 1.



Uniformly shaded strips, 5 strips.   Total width = 55 dots; strip width = 11 dots.

Figure 2.



```
Fetching snapsdir ( 6:13:09 PM )
Storing snapshot-1 ( 6:13:09 PM )
Storing snapshot-2 ( 6:16:17 PM )
Storing snapshot-3 ( 6:20:28 PM )
```

```
total 24
-rw-        1 nsv           550 Mar 21 14:00
                                          SendMail_Edit.CKP
-rw-        1 nsv         17732 Mar 12 16:06 graphhelp.d
-rw-        1 nsv           534 Mar 11 11:30 helpdraft.d
d           2 nsv          2048 Mar  5 13:54 ibm
d           2 nsv          2048 Mar  5 13:54 sun
% graph
s^@^@^@^@^@^P^@^Pe%
% cd ibm
```

1/2, 1, and 1/2. Similarly the three-step staircase function is a unit single slit of width 1/5 convoluted with five delta-functions located at -2/5, -1/5, 0, 1/5, and 2/5 with respective heights of 1/3, 2/3, 1, 2/3, and 1/3.

If the Fraunhofer amplitude is calculated with the convolution theorem, the result is obtained as the product of a single-slit diffraction factor and an interference factor.

(a) Two-step staircase aperture function

```
x=x+d
u=sin(x/6)/(x/6)
v=1+cos(x/3)
y=(uv/3)^2
```

Use y=(uv/2)^2 for a normalized plot.

(b) Three-step staircase aperture function

```
x=x+d
u=sin(x/10)/(x/10)
v=1+(2/3)(2cos(x/5)+cos(2x/5))
y=(uv/5)^2
```

Use y=(uv/3)^2 for a normalized plot.

```
x=x+d
u=sin(x/10)/(x/10)
v=1+(2/3)(2cos(x/5)+cos(2x/5))
y=(uv/3)^2
```

Single slit: Three-step staircase aperture function

Figure 3.

COMPUTER GRAPHICS UTILITIES IN SPECIAL RELATIVITY

Presenter: Edwin F. Taylor *Massachusetts Institute of Technology*

ABSTRACT

Special relativity describes phenomena remote from common experience: the world of the very fast. Graphical computer models display surrogate worlds that students can explore and manipulate to help them visualize the real one. Student use of these models is assured by embodying them in utilities needed to solve exercises in homework assignments and take–home examinations. We describe the use of such interactive computer graphics utilities with a class of 51 students in special relativity at the Massachusetts Institute of Technology. Most students were enthusiastic about the programs and many remarked on the help they gave them in visualizing the subject.

PURPOSE

We want students to develop knowledge of and intuition for special relativity. We want them to be able to analyze real and possible observations and experiments for which relativity is the sovereign theory.

The major difficulty in this task is that the world of relativity is far from common experience. Most speeds observed in everyday life are much less than that of a jet airliner, which moves at approximately one millionth of the speed of light. Playing with balls, blocks, and wagons when we were children trained our muscles in Newtonian mechanics. No such encounters with high–speed objects prepare our intuitions for the results of special relativity.

The computer can model the world of the very fast, providing a "playground" for developing intuition in special relativity. For example, a visual display can show what the world would look like if the speed of light were 50 miles per hour. The visual experience of traveling through such a model world is satisfyingly weird, with strange distortions in our view of objects as we pass them (Figure 1) and violent changes in their apparent color and brightness (not shown in the present displays).

The fundamental result of relativity is that spacetime is truly four–dimensional, with three spatial dimensions and one temporal. To see the four–dimensional character of spacetime we adopt the same strategy that makes a landscape take on meaning: we look at it from different points of view. In relativity the "points of view" are different reference frames in relative motion. In each of these frames the spatial dimensions are probed by rods and the time dimension by clocks. The computer can picture rods, clocks, and light flashes from the viewpoints of various observers in relative motion (Figure 2).

More abstractly, special relativity locates events in the four–dimensional world of spacetime. No more graphic demonstration of the unity of spacetime exists than the t vs x plot of events and the transformations of this plot among frames in relative motion (Figure 3). All possible cause–and–effect relations among events occurring along one spatial dimension are reflected in this plot.

However, the "practical payoff" of the theory of special relativity comes from its use as a work–a–day toolbox for the analysis of cosmic events and of interactions among sub–atomic particles. Graphical plots of energy vs momentum embody the unity of energy and momentum for interacting particles (Figure 4). This unity plus the conservation of total energy and total momentum allows the computer to help predict the outcomes of many interactions.

Computer programs are useful only if they are used. Students gave only passing attention to our demonstration and simulation programs. In contrast they made heavy use of interactive graphical utilities that helped them analyze physical systems investigated in homework and examination problems.

In summary, our overt goal in using computers in this class is to empower the student with utilities that model relativistic laws graphically, reduce tedium, speed solutions, allow analysis of more difficult cases, and apply relativistic theory to as wide a range of physical systems as possible. Our covert goal is to deepen visualization of and intuition for the great central themes of special relativity.

THE SETTING

The Massachusetts Institute of Technology is exploring computers in education, an effort internally administered under Project Athena. The goal of Project Athena is to integrate computers with education throughout the curriculum.

Development efforts in the School of Science utilize personal computers donated by the International Business Machines Corporation. In addition, the Project awarded to our particular effort a series of grants providing wages for student programmers, most of whom are either physics majors or freshmen and sophomores intending to become physics majors.

In the fall of 1985 fifty-one students (63% sophomores, 61% physics majors) in an elective MIT class in special relativity used the graphics programs described briefly above (and in more detail below) in solving one third of the homework exercises and one take-home examination out of three examinations. This course is what MIT calls a "nine credit class" (instead of the regular 12 credits): it meets twice a week for a one-hour lecture. The text was Spacetime Physics by Taylor and Wheeler (W. H. Freeman and Co.) and supplementary reading was taken from Relativity Visualized by Lewis Carroll Epstein (Insight Press) and from class handouts. Evaluation of student performance was based on weekly homework assignments, weekly reading quizzes, and three examinations, one of which was a take-home examination making heavy use of the computer utilities.

In order to assure accountability for computer-aided solutions, the students typically turned in a dot-matrix graphic printout of the computer screen showing these solutions. They were required to label and annotate this printout by hand to demonstrate their understanding of the assignment and of their own analysis. A classroom video projector helped introduce the computer utilities and served as an animated blackboard.

COMPUTER FACILITIES AND PROGRAMS

We developed these programs on three machines: an IBM PC, an XT, and an AT, the latter provided by Project Athena. Students had access to an Athena cluster of IBM AT computers, four with dot-matrix printers. The graphics utilities provided to this class were "finished" products; student users did no programming. About 80% of the students said it took them an hour or less to master the computer facility for this class. Seven students estimated zero time or a few minutes, and only three said they required two hours or more. Students were generally pleased with the availability and maintenance of the machines, although several expressed a preference for computers in the dormitories, which is part of Project Athena's ultimate goal. A planned network for IBM machines is not yet in place, so each student received a self-booting floppy disk containing the programs. Halfway through the term this was replaced with a corrected and updated disk. None of the programs write to soft or hard disk; the only output is the graphic printout handed in with homework problems.

The programs are written in the languages C and PASCAL using a variety of graphics packages: Halo Graphics, Metawindow, and more recently GKS. Development was part-time for faculty and an average of two student programmers during the academic year 1984-85, full time during the summer of 1985, and part-time again when the class was offered in the fall semester 1985.

Here are somewhat fuller descriptions of the programs, which were developed by various student programmers.

VISUAL APPEARANCE (Figure 1), developed by Thomas LeCompte, is a simulation showing the one-eyed view of a rudimentary landscape seen through the windshield of a rocket ship moving with any speed up to that of light. The landscape consists of four stick figures: a large cube, a small cube, a pyramid, and a skyscraper. The operator can accelerate the rocket, decelerate, turn right or left, and change altitude.

OBSERVER (Figure 2), developed by Eric Berman, is a more formal and quantitative representation of space and time as observed in laboratory and rocket frames of reference. Numbered coordinates and a string of digital clocks in each frame demonstrate the effects of time stretching, Lorentz contraction, and relative synchronization of clocks. The observer can monitor events from either the laboratory or rocket frame of reference. A shuttle, positioned between the two frames and having its own independent speed, emits flashes showing Doppler shift effects and can be made to execute a round trip to demonstrate the Twin Paradox.

SPACETIME DIAGRAM (Figure 3), developed by Eric Berman, displays the locations of events in space and time. The x-position and time of an event are plotted as horizontal and vertical screen coordinates respectively. The operator can create event-points, connect them with world lines, construct through them light cones and invariant hyperbolas, and transform them to other reference frames moving with respect to the original frame (among other capabilities.)

ONE-DIMENSIONAL COLLISIONS (Figure 4), developed by Shawn Gaither, helps the student to solve and display the interactions of sub-atomic particles and photons moving in one spatial dimension: collisions, creations, transformations, decays, and annihilations. The operator enters on a table what s/he knows about the masses, energies, and momenta of incoming and outgoing particles. The computer then calculates the remaining entries using standard conservation laws and other relations, reporting to the operator each logical step in the solution. The results are displayed in the table and plotted on momentum–energy diagrams before and after the interaction. A movie shows an animation of the event along with spacetime and momentum–energy representations. The operator can view the interaction from the rest frame of any particle (except the photons) or from the analytically–important zero–momentum frame in which the momenta of all the particles add up to zero.

NEWTONIAN versions of OBSERVER and SPACETIME DIAGRAM employ the Newtonian model of space and time (incorrect at high speeds) in which time has the same value for all observers and particles and reference frames can move with respect to one another faster than the speed of light. Out of fear that these programs would teach the wrong things, I made no assignments requiring them. As a result they were little used by students.

RESULTS

The consequences of computer use in this class are being studied in collaboration with Dr. Karen Cohen of Project Athena, who observed the class, tabulated student computer logs, interviewed groups of students, helped to design a student questionnaire and study its results, and compared outcomes in this class with those of other classes using computer graphics at MIT. The ten–page questionnaire, filled out on the last day of classes by 40 of the 51 students enrolled, sought reactions to many aspects of the course with emphasis on the computer graphics. Preliminary results of this study follow under several headings.

HEAVY USE

The computer programs were heavily used. The handed–in homework included the required annotated graphics printouts. The third examination, a take–home exam resulting in a high class average, required a quantity of results that would have been overwhelming without computer assistance. Students reported that they used the computers an average of 1.5 hours per week out of a total average of 3.2 hours per week spent on homework.

UTILITIES POPULAR

The most-used programs were the utilities: SPACETIME DIAGRAM and ONE-DIMENSIONAL COLLISION. This preference is not surprising, since the utilities were needed to carry out the assigned exercises. However, student enthusiasm was also greatest for the utility programs. Here are some quotes from the questionnaire.

SPACETIME DIAGRAM:

"taught Lorentz transformations very well. A lot of situations could be simulated on it."

"Great! Really helps you to 'see' what is happening in a problem."

ONE-DIMENSIONAL COLLISION:

"Helps much to visualize and perform calculations which would be difficult by hand."

"This cleared up many of my questions about collisions in particle physics."

OBSERVER, part way between a simulation and a utility, was less popular than the straight utilities, although the amount of use varied widely among students. This was the first program we wrote, and our inexperience was reflected in the response of some students to the graphics.

"A good 'stepping stone' program to introduce time dilation, etc."

"A little hard to interpret, but good once you get used to it."

"Screen is too busy. Hard to tell what some of the little lines mean after a while."

VISUAL APPEARANCE, an almost pure simulation, was little used by students, and then mostly for "playing around."

"Cute to look at — didn't heighten understanding"

"Seemed more like an interesting demonstration than a learning tool."

TEDIUM REDUCTION AND AID TO UNDERSTANDING

We were happy to see that some of our original goals were apparently accomplished, including usefulness of the programs for solving the exercises and for understanding the subject.

"[The programs] made many tedious calculations quickly, and this gave a fairly quick access to answers of many would–be 'tedious' problems. They also made many problems, which would otherwise be difficult or impossible to solve, easy to do and easy to visualize."

"The computer could grind through boring math and ugly square roots, so you, if you wished, could think about the physics."

"Having transformations done instantly was very helpful in gaining intuition."

"[Computer programs] helped me to develop an intuitive feel for relativity."

"It is the main thing that made me understand the material."

"VISUALIZING" THE SUBJECT

We were surprised by the number of times that the word "visualize" and its variants appeared spontaneously in questionnaire results and group interviews.

"[Computer programs] Made some of the material much easier to visualize."

"I found it helpful in visualizing concepts which are quite hard to visualize otherwise."

"[The computer helped give] Insight into the nature of the physical phenomena associated with relativistic particles and aid in the visualization of such phenomena."

"Like the SPACETIME program, [the COLLISION program] helped me to visualize mass/energy/momentum."

For some students, at least, we appear to have achieved our covert goal of encouraging intuition and visualization of the central themes of relativity. In addition, the "visual accountability" promoted by using graphics utilities and handing in printed graphics outputs is unusual in physics courses, and we welcome it.

As instructor, I was also affected by the interactive graphics displays. They changed my own style of envisioning and presenting relativity. Contributing to this change of style was the "animated blackboard" provided by a portable projection video and computer on a cart, which allowed quick classroom demonstration of many effects under discussion. Of course my immersion in developing the displays influenced me more than merely using them would influence other teachers. John Archibald Wheeler and I are now rewriting the text used in this course with fewer equations and more graphics. This may make the book accessible to a wider audience, whether or not computer software is included.

STUDENT COMPLAINTS

Almost all the students complained about the slowness of the printer used to print out the screen displays handed in with homework.

"The printer took a lot of time and the computer had to remain idle during that time."

"PRINTERS take *?!@ long time and keyboards freezup [sic] -- takes 20 minutes for 3 printouts."

"the printers are terrible."

"The printers were much, much, much too slow!"

We expect to set up a buffer system to allow printing while students do other exercises on the machine.

The students were issued ONE-DIMENSIONAL COLLISION before all bugs were removed from the program. Although no student complained that residual difficulties seriously hobbled their performance on homework or examination, tolerance was low for any bugs whatever.

A substantial number of students worried that the ease of computer use can lead to superficial understanding of the subject.

"Too, too easy to substitute computer symbol manipulation for physical intuition. WATCH OUT."

"Sometimes computers were used to the exclusion of important theory questions."

"--didn't help one learn why something occurred, just how to enter data into an IBM-PC"

Some students with this complaint also had a suggested cure.

"Some pencil work is desirable too!"

"[The SPACETIME DIAGRAM program was] very helpful, but I would have liked to do some diagrams myself First ."

"I would have liked to learn how to do what the computer is doing more thoroughly before letting the computer do it....Computer programs are good, but there's a tendency to not understand what exactly it's doing, and how it comes up with the results it does."

"I would replace some of the computer work with more analytic, hand-solved problems. The computer doesn't inspire confidence in problem-solving ability."

A student who expressed approval of the balance in computer use in relativity provides a comparative glimpse of computer-assisted education in different classes at MIT.

"I had two classes which used [Project] Athena [facilities] this term. In my other class I felt the computer was used so heavily it hurt my understanding of the material. I applaud the balance between computer problems and problems to do by hand in [this class]."

On balance, I feel that the student worry about dependence on the computer is justified, even though (1) new educational tools may legitimately change the balance of different activities in learning, (2) everyone is uneasy in the face of change, (3) students sometimes hold conservative attitudes toward educational innovation, and (4) two-thirds of the exercises and examinations did not make use of computer utilities. In a nutshell, the physicist needs to feel confident in his or her ability to analyze a problem without the computer. Next fall I will try the student-suggested prescription: assigning some introductory computer-style handwork homework before the computer is enlisted to do the same tasks faster and to take the student into more difficult territory.

Evidently anxiety about "computer superficiality" has little effect on student satisfaction with the amount of computer use in this class. When asked "Overall, would you like to see the use of computer in this course increased, kept the same, or decreased?" 27 chose "just the same," 4 "increased", and 8 "decreased."

OVERALL RESPONSE

There were no global criticisms of computer use in this class, no student who expressed the feeling that the whole effort in relativity was a waste of time. The vehemence of comments about slowness of the printers gives some assurance that students did not frame their general reactions so as to protect our feelings. Here from the questionnaire are what I judge to be the most negative general comments on computer use in the class, excluding those already quoted, the large number of complaints about slow printer speed, and a smaller number on software crashes.

"All prog[ram]s — We need more displayed accuracy! Also needed are finer transform step sizes!...It does not increase confidence in the student's ability to solve problems by hand."

"it made things sometimes redundant."

"Emphasis upon computation rather than theory — actually, it is too much of a good thing."

"Too [sic] simple"

"Computers are simply less accessible than my desk. Hardly their fault."

A much larger fraction of the general comments were enthusiastic. A few of my favorites:

"It was a wonderfully rich method for teaching some otherwise difficult concepts. A real breakthrough!"

"An easy and graphic visualisation [sic] of what we were doing as well as being a fast way to do tedious things such as Lorentz transforms and collisions and pair production."

"It is easier to concentrate on conceptualizing the physics and understanding the solution instead of getting lost in calculations."

"Easy to try out things just for interest."

"These programs are a wonderful breakthrough in teaching this subject and probably are a forerunner for use of computer assistance in other areas of physics teaching."

CONCLUSIONS AND PLANS

We feel confirmed in our intuition that computer graphics utilities provide an educational payoff, at least in relativity. Student performance and comments seem to show that utilities assist student understanding and visualization of the subject, allow analysis of a greater variety and more difficult physical systems, take advantage of student preoccupation with performance on homework and examinations, and promote "visual accountability."

Current efforts pursue this theme. The original OBSERVER was more demonstration than utility and not sufficiently versatile in analyzing thought experiments. It is being completely rewritten.

The original COLLISION was limited to one spatial dimension, with energy as the second displayed dimension. A majority of high–energy particle interactions of interest to physicists can be analyzed in two spatial dimensions. A new COLLISION program assists in the analysis of such interactions. Showing two momentum components plus energy requires three dimensions, and this complicates the display. Rotation of perspective views aids visualization. Figure 5 does not show the range of colors on the screen that contributes significantly to this visualization.

Of course any new teaching aid, particularly a successful one, can lead to preoccupation with a single component of education. Human interaction remains the basis of teaching and learning. Even in those aspects appropriately assisted by the computer, a mix of different kinds of programs can serve a variety of purposes. Among our programs, VISUAL APPEARANCE is the one most often shown to visitors (and was the first one described in this paper) because it mimics direct experience. Even though VISUAL APPEARANCE is not a utility and students soon move on to other programs, we employ it as businessmen do a "loss leader," a product they advertise below cost to attract potential customers into the store, where more profitable transactions can take place.

SOME PRACTICAL SUGGESTIONS

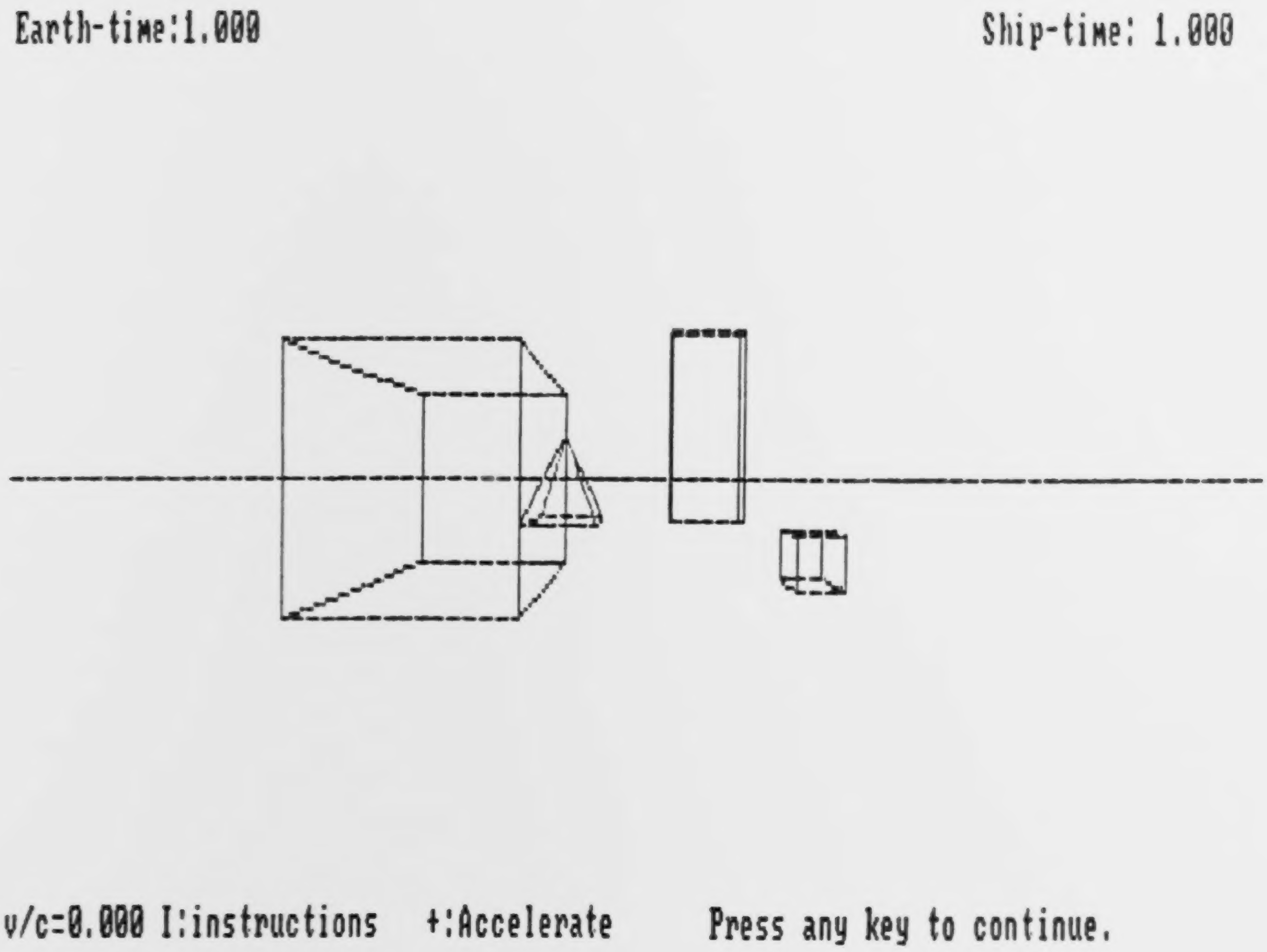
It takes hundreds of hours to create, modify, and debug a graphics utility. Student programmers are the key component in this project. They learn a lot while providing an economical service. Academic developers should place their major financial resources in student wages.

Technical advice matches the dreams of the teacher to the ever–shifting world of the dependably possible. In our experience the most difficult technical task is vertical compatibility: combining all the pieces of hardware and software to provide a transparent and trouble–free facility. The staff and students of Project Athena provided this essential ingredient.

Scheduling, access, maintenance, advice, instruction sheets, troubleshooting... Class organization for the computer is much more demanding in these crucial minutiae than paper technology. In practice you will hear about them only when they do not work, and by then it may be serious. These matters are not mentioned much in this article, partly because they go without saying, partly because we and Project Athena did most of them right — except for the slow printers!

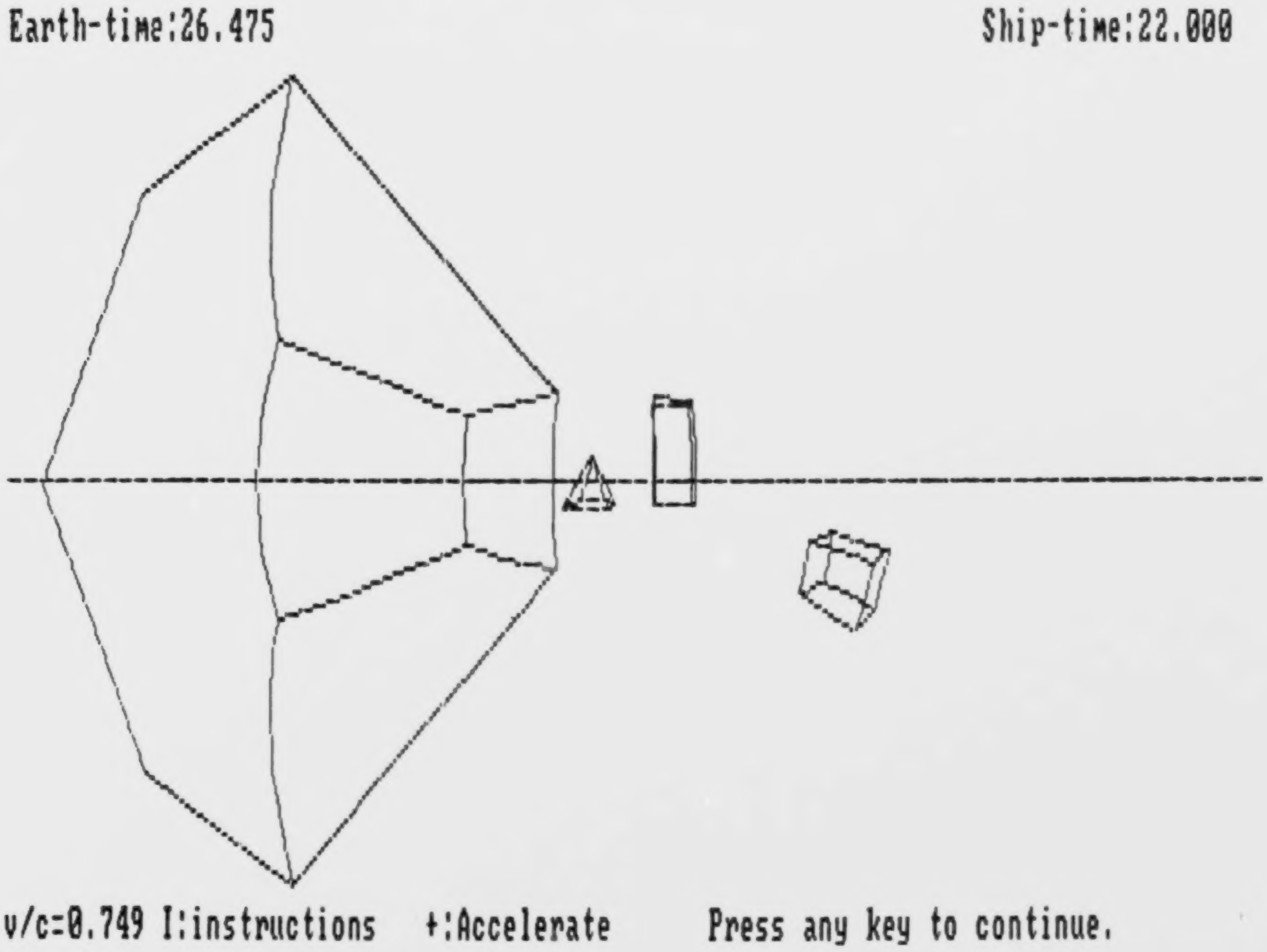
ACKNOWLEDGMENTS

Student programmers Eric Berman, Thomas LeCompte, and Shawn Gaither were the backbone of the project reported here. They combine breathtaking technical skill with an enthusiasm that renews their colleagues. A committee chaired by Professor James L. Kinsey provided steady support and strategic advice. Detailed help from many people at Project Athena turned a slow, amateur educational development into a faster, more professional one. Karen Cohen was particularly helpful in seeking out significant results of a complex and fascinating human endeavor. We thank the International Business Machines Corporation for their support of a larger effort of which the present project is a part. We are grateful to students in the relativity class for their willingness to try something new and to offer thoughtful suggestions. Without their participation this work would not be possible.



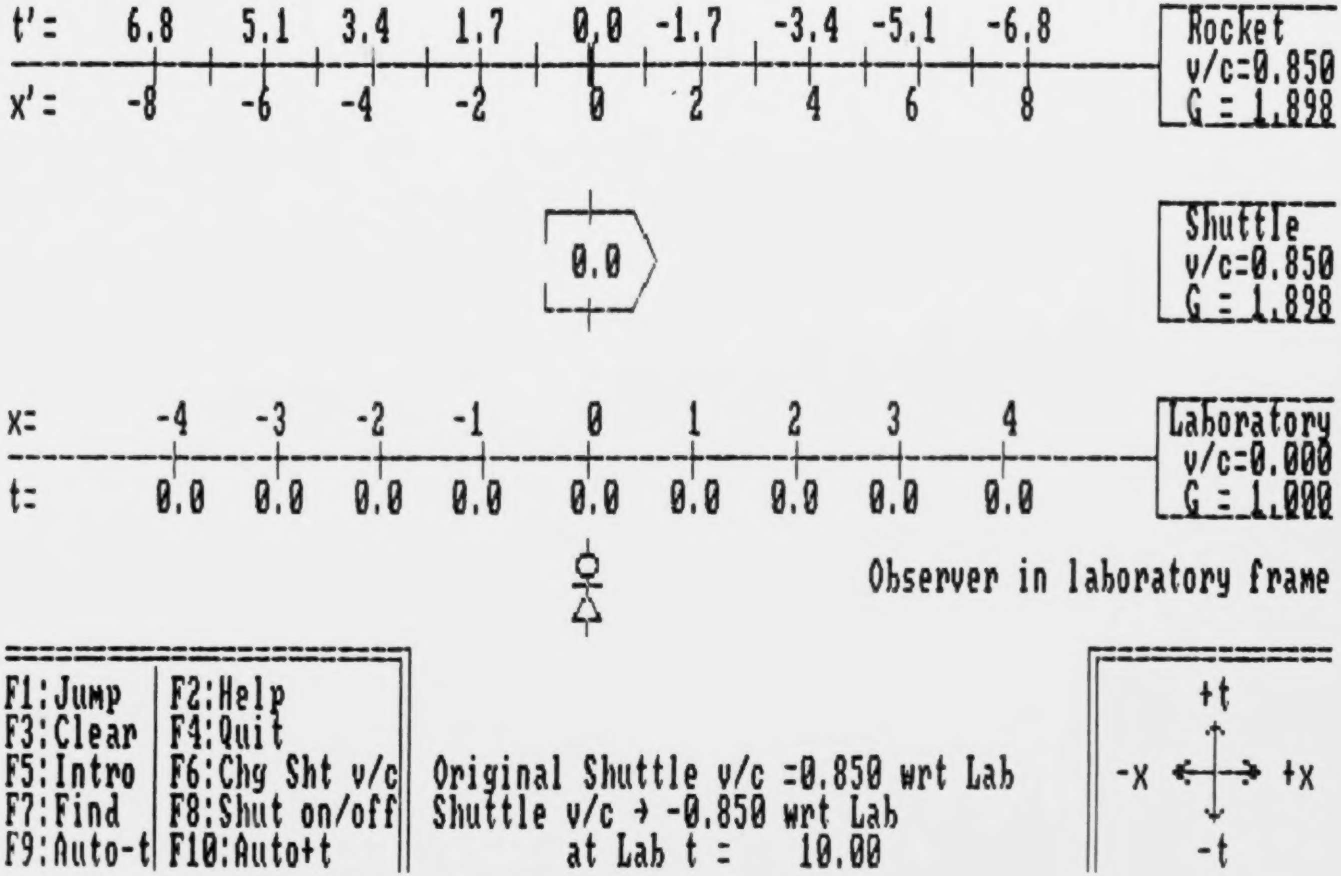
```
Earth-time:1.000                                  Ship-time: 1.000

v/c=0.000 I:instructions   +:Accelerate     Press any key to continue.
```

Figure 1.a
VISUAL APPEARANCE display. View forward through the windshield of a rocket ship at zero velocity.



```
Earth-time:26.475                                          Ship-time:22.000

v/c=0.749 I:instructions   +:Accelerate      Press any key to continue.
```

Figure 1.b
VISUAL APPEARANCE display. Rocket view at 3/4 the speed of light. The large cube appears to turn partly "inside out," the small cube has apparently rotated, and the pyramid and skyscraper appear to be farther from the observer than orginally.



Rocket
v/c=0.850
G = 1.898

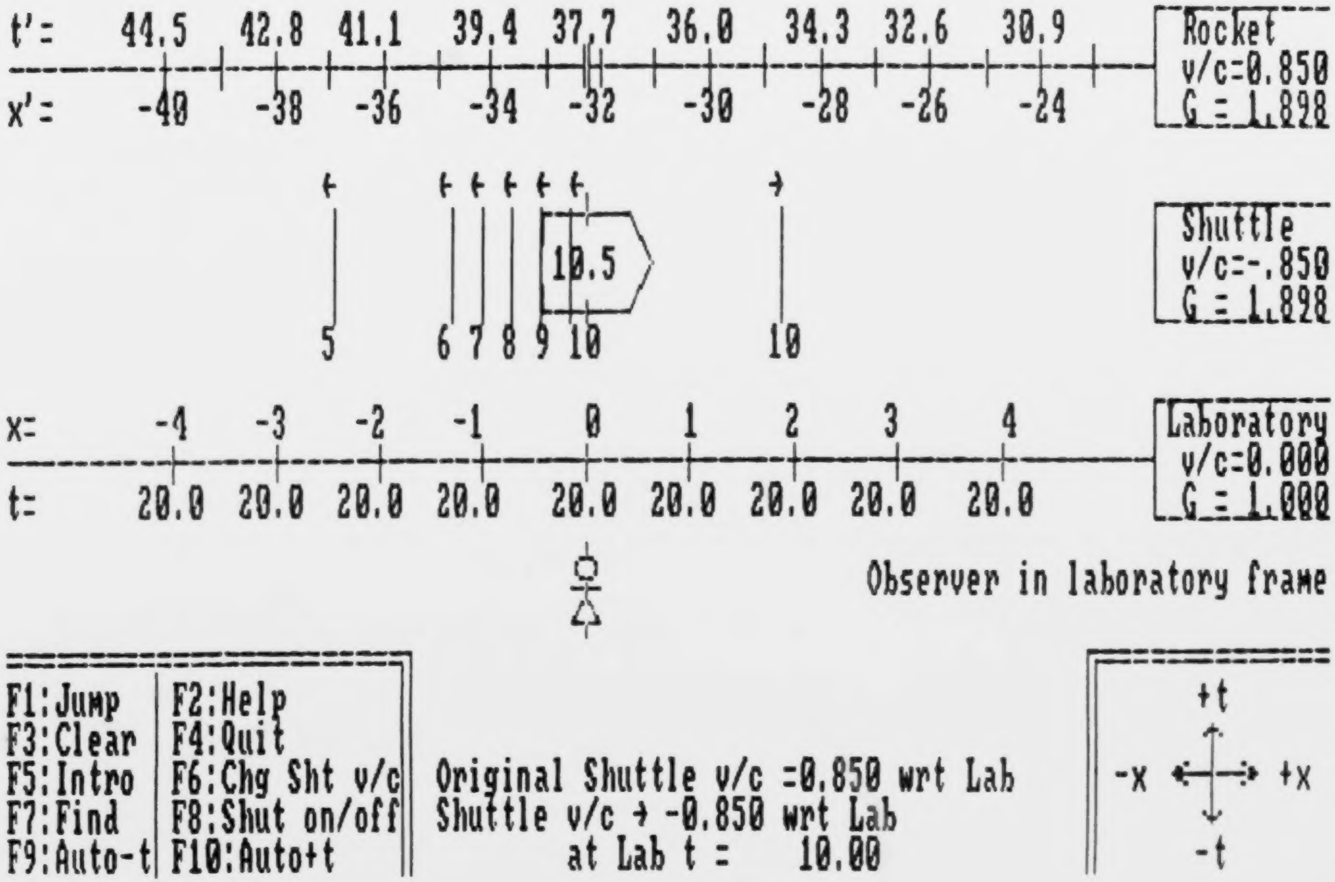
Shuttle
v/c=0.850
G = 1.898

Laboratory
v/c=0.000
G = 1.000

Observer in laboratory frame

Original Shuttle v/c =0.850 wrt Lab
Shuttle v/c → -0.850 wrt Lab
at Lab t = 10.00

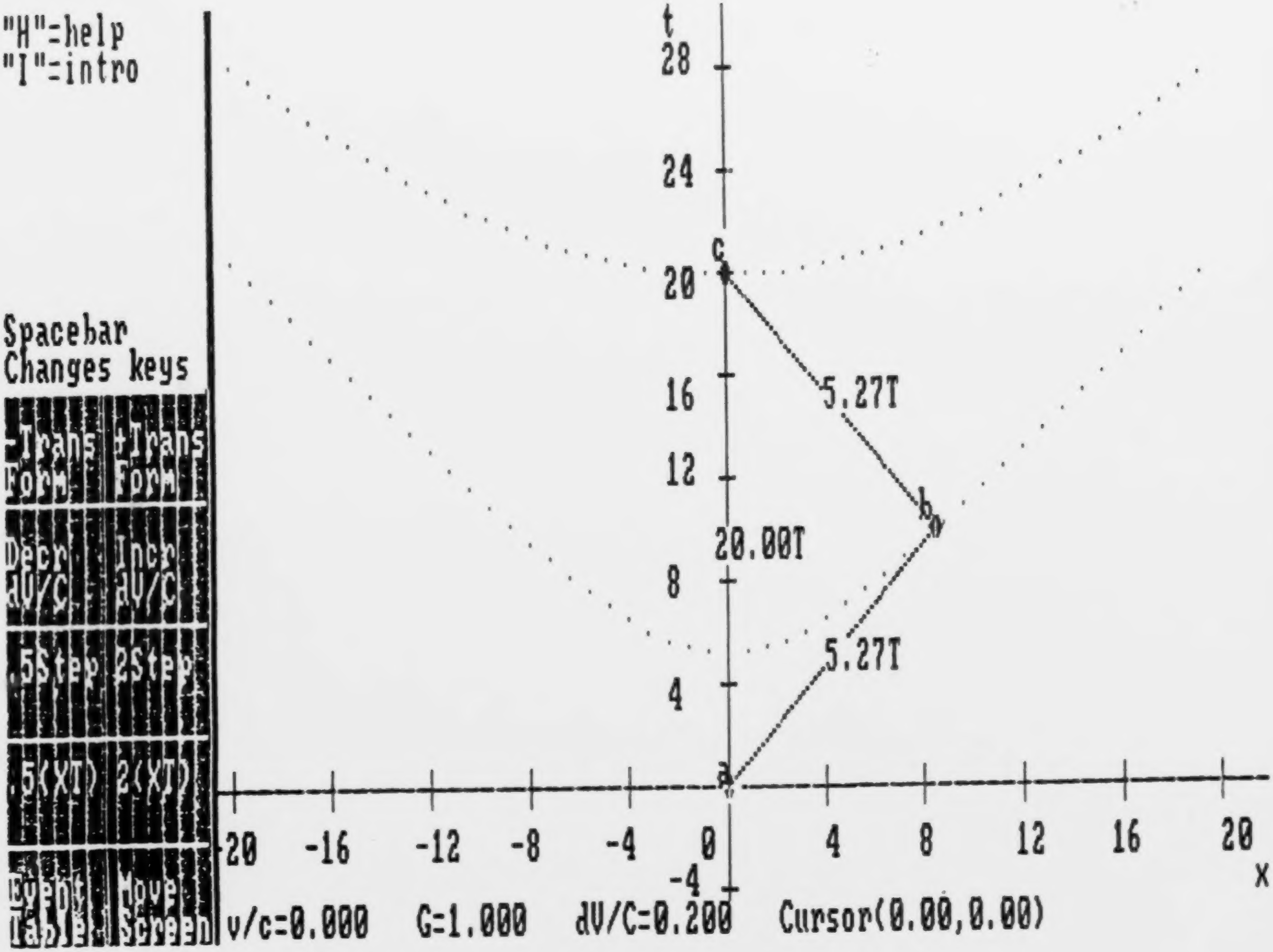
Figure 2.a OBSERVER display at laboratory time zero.



Observer in laboratory frame

Original Shuttle v/c =0.850 wrt Lab
Shuttle v/c → -0.850 wrt Lab
at Lab t = 10.00

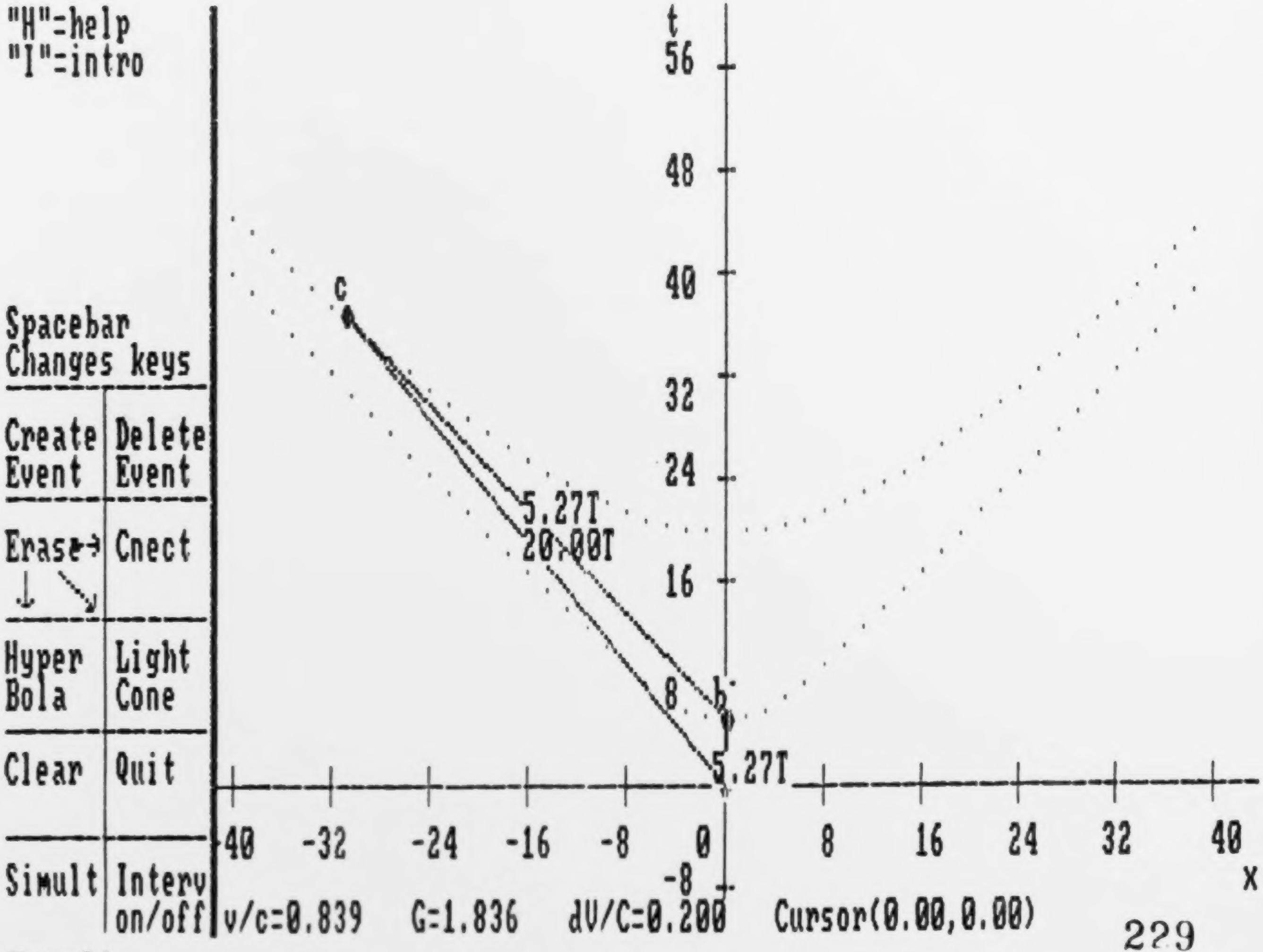
Figure 2.b
OBSERVER display after shuttle has made a round trip. Notice that the shuttle clock reads 10.5, whereas the laboratory clocks read 20. This is the famous "Twin Paradox".



v/c=0.000 G=1.000 dV/C=0.200 Cursor(0.00,0.00)

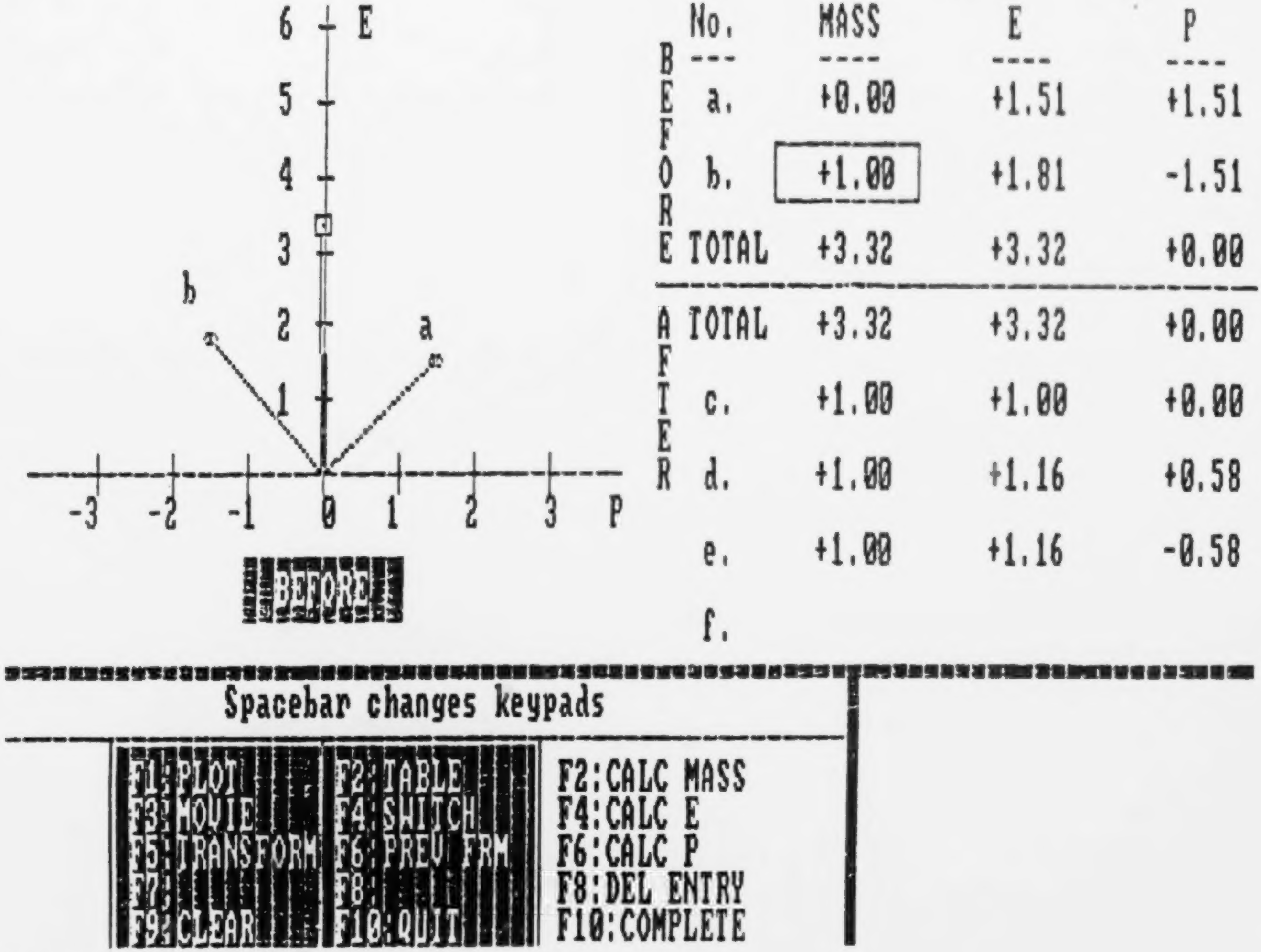
Figure 3.a

SPACETIME DIAGRAM display of the shuttle round trip. X-displacement is plotted horizontally, time vertically. "Clock times" are plotted along the world lines. Notice that the sum of outgoing and returning clock times for the shuttle is less than the time along the vertical world line of the "no-motion" stay-at-home twin.



```
v/c=0.839   G=1.836   dV/C=0.200   Cursor(0.00,0.00)
```

Figure 3.b
SPACETIME DIAGRAM of the same round trip as observed in the referenced frame of the outgoing shuttle. Time read on clocks traveling between given events does not depend on their state of motion.

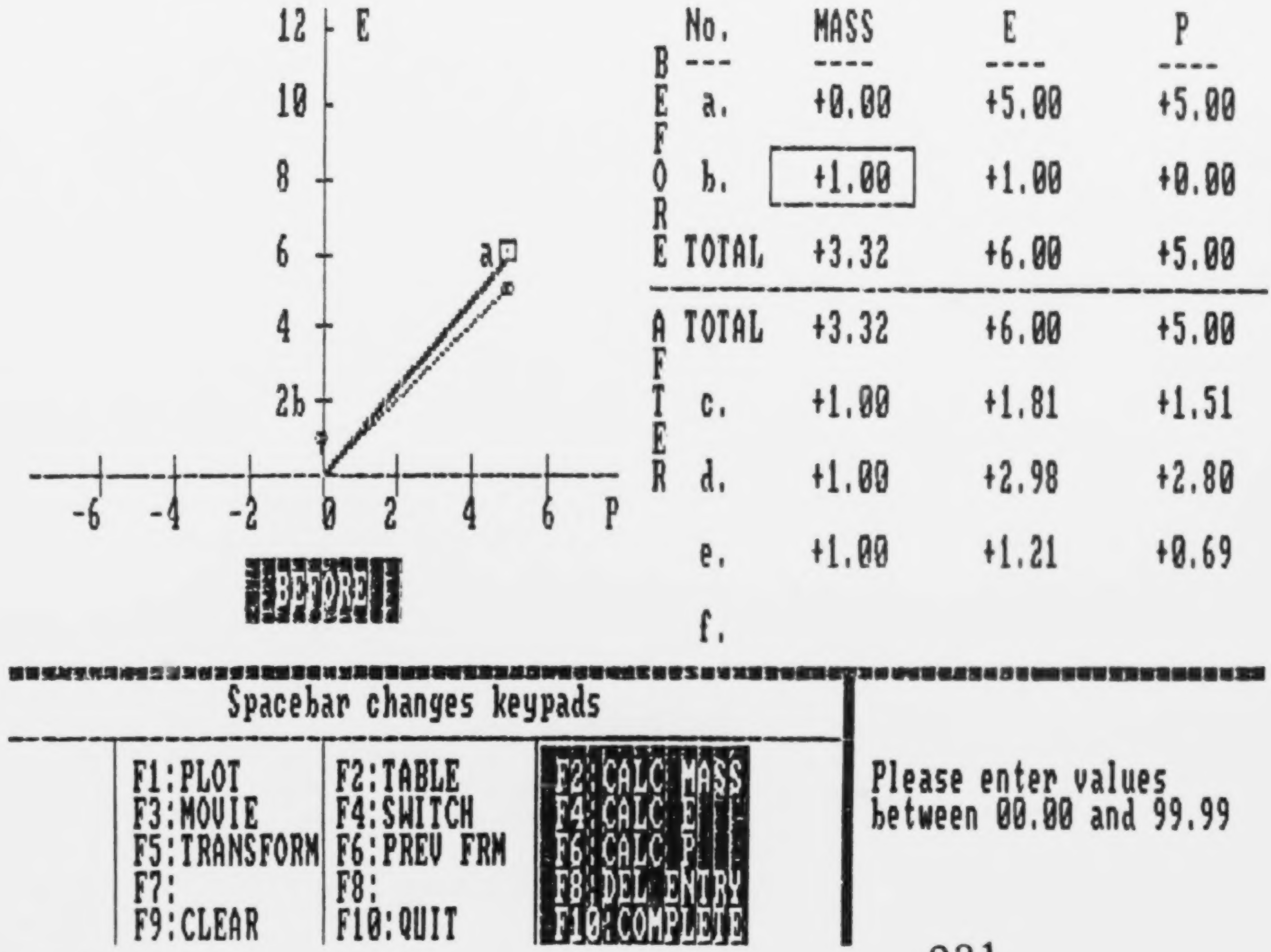


```
     No.     MASS       E        P
B    ---     ----      ----     ----
E     a.     +0.00     +1.51    +1.51
F
O     b.    [+1.00]    +1.81    -1.51
R
E  TOTAL     +3.32     +3.32    +0.00
------------------------------------------
A  TOTAL     +3.32     +3.32    +0.00
F
T     c.     +1.00     +1.00    +0.00
E
R     d.     +1.00     +1.16    +0.58

      e.     +1.00     +1.16    -0.58

      f.
```

```
        Spacebar changes keypads
-------------------------------------------------------
   F1:PLOT        F2:TABLE       F2:CALC MASS
   F3:MOVIE       F4:SWITCH      F4:CALC E
   F5:TRANSFORM   F6:PREV FRM    F6:CALC P
   F7:            F8:            F8:DEL ENTRY
   F9:CLEAR       F10:QUIT       F10:COMPLETE
```

Figure 4.a

ONE-DIMENSIONAL COLLISION display. A gamma ray (zero mass) strikes an electron (mass unity) to create a positron-electron pair. Here the interaction is observed from the "zero-momentum frame," in which total momentum is zero. Total energy and momentum of the system is plotted as a small square.



	No.	MASS	E	P
BEFORE	a.	+0.00	+5.00	+5.00
	b.	+1.00	+1.00	+0.00
	TOTAL	+3.32	+6.00	+5.00
AFTER	TOTAL	+3.32	+6.00	+5.00
	c.	+1.00	+1.81	+1.51
	d.	+1.00	+2.98	+2.80
	e.	+1.00	+1.21	+0.69
	f.			

Spacebar changes keypads

Please enter values between 00.00 and 99.99

Figure 4.b
ONE-DIMENSIONAL COLLISION display. Same event as observed in a frame in which the initial electron is at rest.

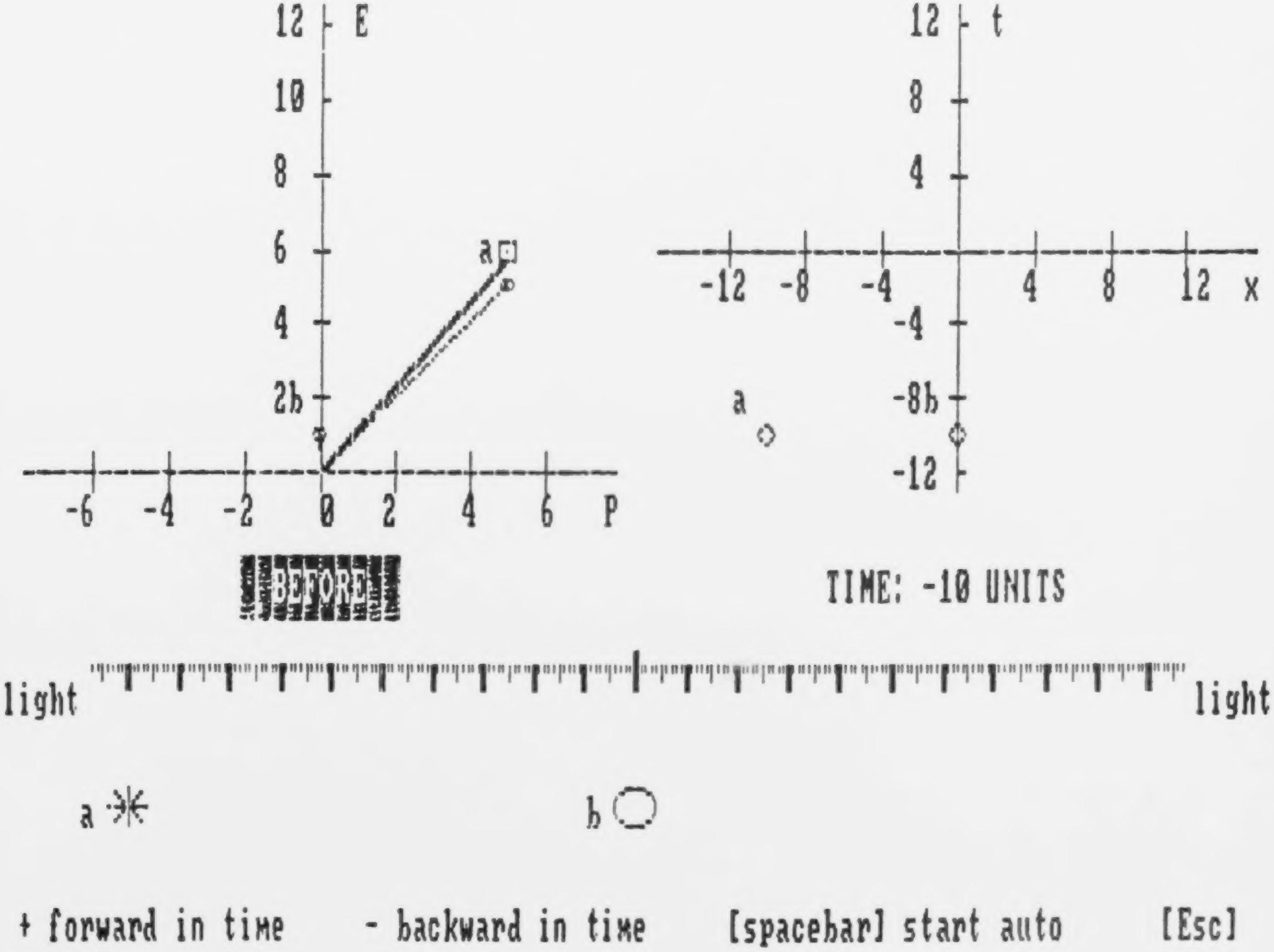


Figure 4.c

ONE-DIMENSIONAL COLLISION movie showing particles before the collision described above. Movie is at the bottom. Spacetime diagram, upper right, plots particle trajectories in spacetime. Momentum-energy diagram, upper left, shows particle momenta and energies before the collision.

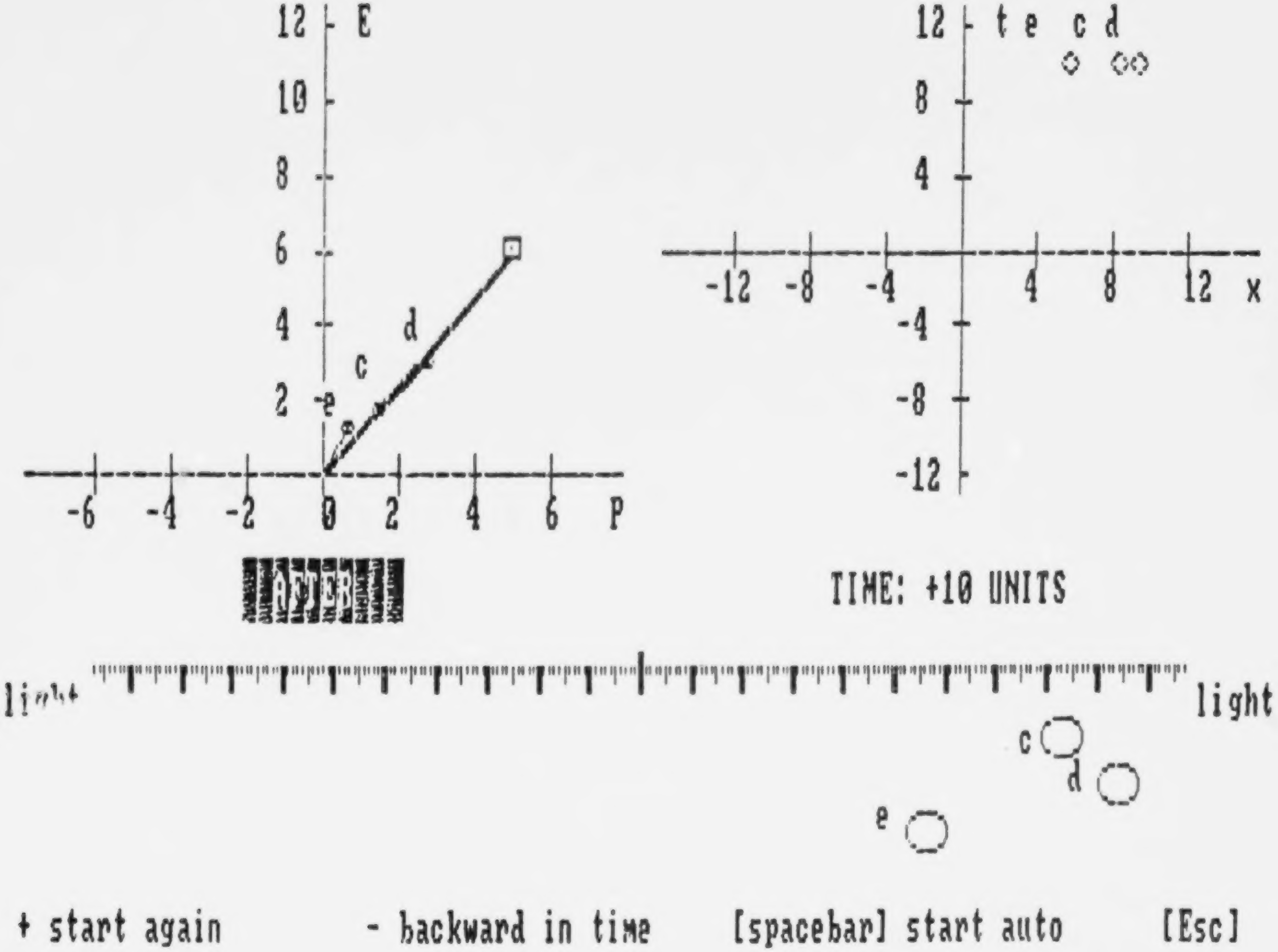
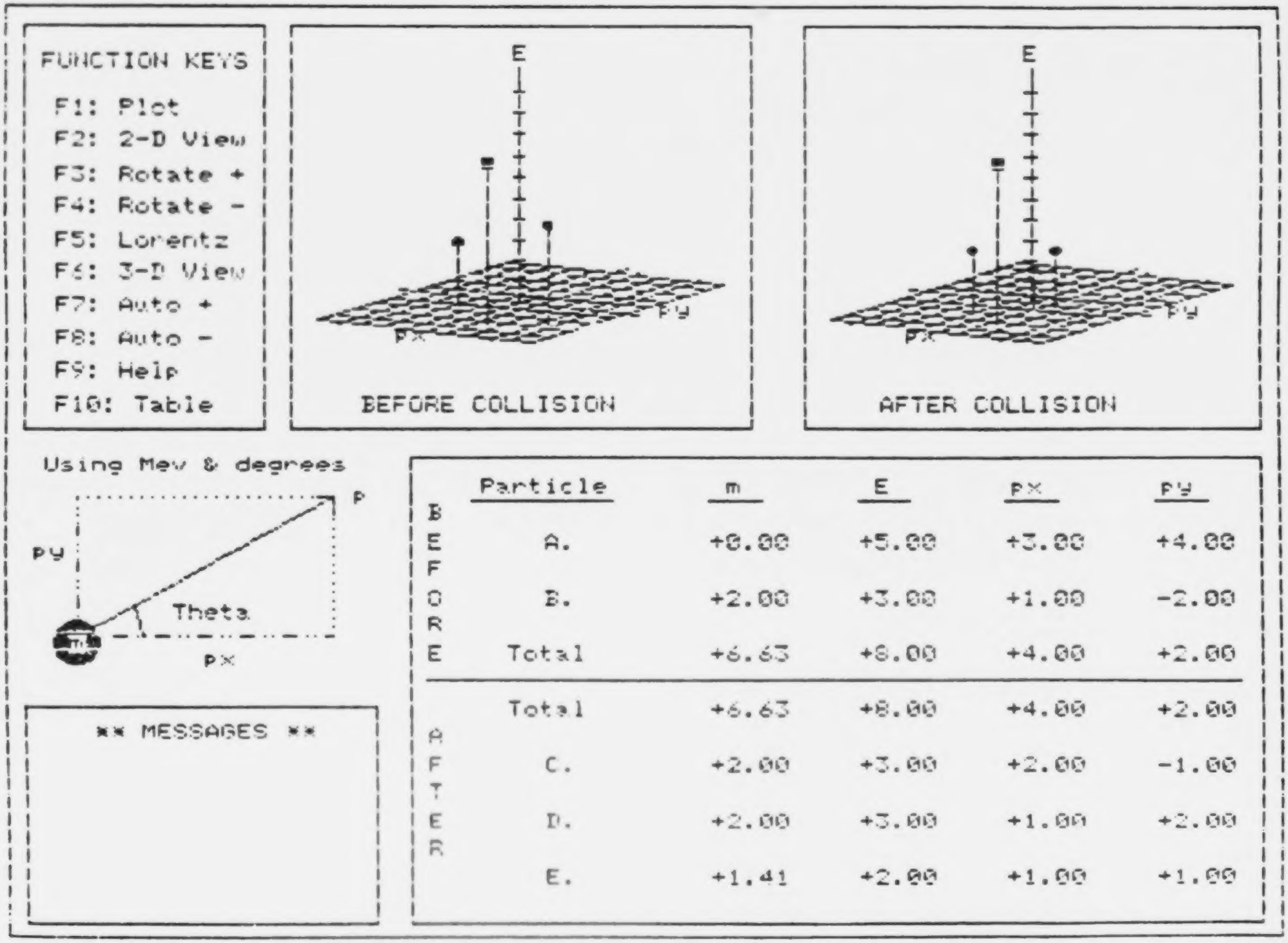


Figure 4.d
ONE-DIMENSIONAL COLLISION movie showing particles after collision described above. Now there are three particles: the original electron plus an additional positron-electron pair.



	Particle	m	E	Px	Py
BEFORE	A.	+0.00	+5.00	+3.00	+4.00
	B.	+2.00	+3.00	+1.00	-2.00
	Total	+6.63	+8.00	+4.00	+2.00
AFTER	Total	+6.63	+8.00	+4.00	+2.00
	C.	+2.00	+3.00	+2.00	-1.00
	D.	+2.00	+3.00	+1.00	+2.00
	E.	+1.41	+2.00	+1.00	+1.00

Figure 5.a
TWO-DIMENSIONAL COLLISION display under development for future class use. This dot matrix printout does not do justice to the detail of the screen image. In addition, use of several screen colors improves visualization.

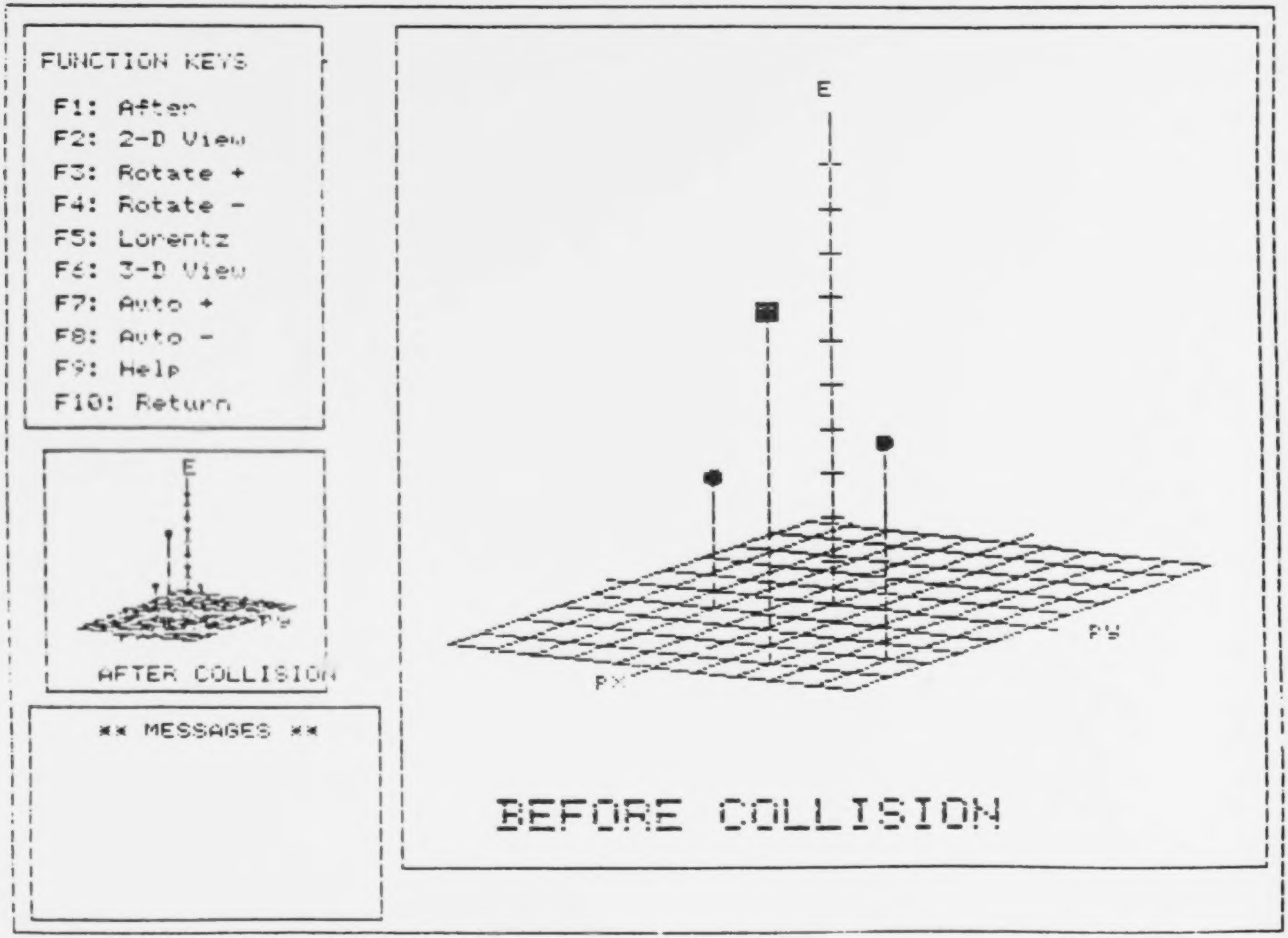


Figure 5.b
TWO-DIMENSIONAL COLLISION display showing magnified plot of momenta and energy before the collision. Total energy and momentum of the system is plotted as a small square. This program also displays a "movie" of the two-dimensional trajectories before and after the collision.

COMPUTER SIMULATIONS OF PLANT MORPHOLOGY AND EVOLUTION

Presenter Karl J. Niklas *Cornell University*

RATIONALE

Most areas in research and teaching have central concepts that involve dynamic processes, such as motion and change in size and geometry. In the past these concepts were communicated either verbally or with a series of static pictures representing various stages in the phenomenon under consideration. The researcher or student was expected to use these descriptions to produce a mental cartoon or filmstrip in which portions of the phenomenon not contained within the pictures or verbal descriptions are reconstructed through intercalative thinking processes. This research or instructional environment is no longer satisfactory, particularly since advances in small computers now provide an opportunity to use computer graphics and simulations as tools to construct visualizations of complex phenomena. In addition, these tools can be augmented to provide numerical solutions to aspects of the phenomenon under consideration, as well as an opportunity to introduce variables that affect the outcome of such computations. The strength of computer graphics lies not so much in "cartooning" but in providing mental experiments whose outcomes are restricted by the physical processes that dictate the system. This is a powerful feature both in the laboratory and in the classroom environment.

The Biological Sciences are no exception to the rule that central concepts have dynamic features. Growth, development, community competition, and evolution are essential concepts to almost all biological levels of inquiry. And the communication of these concepts benefits from computer assisted graphics, particularly when the intellectual or pedagogical objective is to determine the role of critical variables in influencing the outcome. Clearly, however, the biological processes under consideration must be susceptible to quantitative analysis and description if computer graphics are to be used with rigorous intellectual content.

In many respects, plants are ideal organisms for the use of computer assisted evaluation. Plants are essentially sedentary and their shape and structure are dictated by the physical laws and relationships that apply to autotrophic metabolism. As such, plants can be viewed as elegant engineering solutions to the transport of fluids (water and sap), the accommodation of mechanical stresses due to vertical growth, the display of light-intercepting surfaces (leaves) to sunlight, and to their reproductive biology, which in many plants involves the fluid dynamics of pollen-capture (wind pollination).

Computer programs dealing with two features of plant shape, function and evolution will be reviewed in this paper. These are TREE, which shows how plant shape is a compromise between mechanical design and the display of photosynthetic tissues, and MODEL, which shows how the fluid dynamics of airflow dictates wind pollination. Both of these programs were generated from within a research environment in which IBM PC-XTs played a critical role. Hence, both of these programs could be transmitted to a classroom environment without significant changes in technology or intellectual content. The PC is a versatile tool in many contexts for the biologist.

TREE (The Biology of Branching Patterns and Growth)

Most land dwelling plants are constructed from cylindrical axes (stems) that grow at their tips (apical meristems) by adding new cells and by occasionally branching to form two or more growing tips. When young, most plant axes are green over their surfaces due to photosynthetic tissues. In addition, leaves are produced, and in many plants these are the principal sites of light-interception. From the principles of mechanical engineering, the mechanical stresses produced by vertically growing plant axes can be calculated. All that need be specified for these calculations are the shape, size and density of axes, as well as their three-dimensional orientations relevant to the ground's surface. These features of plant shape and orientation can be specified by a student and the computer can be used to build a branching structure in three-dimensional space. Simultaneous to this type of simulation, the computer can calculate mechanical stresses. With this capability the student can compare and contrast which branching patterns produce the greatest or least stress during growth and development.

Similarly, the computer can also calculate the capacity of a branching structure to intercept light. This calculation involves the surface areas that a "tree" projects toward the light as the "sun" moves along its diurnal pathway across the sky. This calculation is complex since it also must take into account the consequences of self-shading, i.e., the extent to which plant axes or leaves shade other portions of the plant. Nonetheless, an IBM PC–XT or AT can rapidly compute this value for any three–dimensional shape.

There must be a reconciliation between the mechanical stresses and the light–interception capabilities of a living plant, since maximization of light–interception brings about high mechanical stresses. The computer can calculate these two features of plant shape and the student can determine empirically the relative efficiency of one geometry versus other possible plant shapes.

TREE can be used to predict which plant shapes maximize light–interception and minimize mechanical stresses, and can use this information to evaluate real trees, bushes, or low–growing herbs found in the local environment. By shifting the diurnal pathway of the sun according to latitude, the student can also "test" the performance of different plants in different regions of the world's flora (tropical, north temperate, and subarctic latitudes).

Another use of TREE involves the reconstruction of plant evolutionary trends. The computer can be made to simulate the entire "universe" of all possible branching patterns and calculate the performance of each shape relative to all others. Based on paleobotanical data (which gives the student the shape of the earliest, most primitive land plant) the student can direct the computer to search for the most parsimonious pathway of morpholo– gical changes that increases plant performance. This pathway of change through the universe of all possible plant shapes is an evolutionary trajectory (change in shape through geological time). The student can then compare the computer–predicted evolutionary sequence to the actual fossil record. Comparisons of this sort are instructive since they show whether or not the major trends in plant evolution are dictated by the processes of light–interception and mechanical stability.

MODEL (The Influence of Fluid Dynamics on Pollination)

Many plants capture pollen traveling in air currents and are therefore wind pollinated. This mode of pollination must be dictated by aerodynamics, since pollen grains are essentially small particles suspended in airflow and the pollen–receptive surfaces of female plant organs are wind–baffling structures. With this background the student can be introduced to the principal concepts of aerodynamics and how these concepts influence the biology of plant reproduction. Essentially, the geometry of female plant organs generates airflow patterns through which airborne pollen grains must travel to settle on pollen–receptive surfaces. By the same token, the physical properties (size, shape, and density) of pollen grains influence the manner by which pollen grains navigate through airflow patterns and therefore they influence the probability of pollination.

The student can use MODEL to quantify the behavior of pollen grains with different physical properties as pollen moves through an airflow environment around a female plant organ. This requires that the direction and speed of airflow currents around the female organ are stored in the computer's memory and that the student select the size and density of the pollen to be modeled. MODEL has stored various files that document the aerodynamic environments around a number of plant organs (grasses, pine cones, the female organs of Mormon–Tea, and a variety of rare and exotic plant species). These environments were determined by means of a wind–tunnel technology that allows for the photography of airflow moving around real plant organs. The data from these photographs have been placed within the memory of an IBM PC–XT as separate files. Each file can be called up separately by a student and examined in detail before a pollen grain type is modeled.

The student can specify the number and location of pollen grains released within the airflow environment, as well as the size and density of the pollen. (Size and shape dictate the motion of pollen within the airflow environment, since these physical features influence the velocity of a particle in response to gravity, drag and the pressure field of airflow.) The computer then calculates the speed and direction of pollen–m– option every 1x10–3 sec. In addition, the computer counts the number of pollen grains that collide with pollen–receptive surfaces on the organ.

By modeling a variety of pollen grain types, of which one type is from the species of the plant organ being examined, the student can compare the pollination–efficiency of the female organ. If the pollen of the same species as the female organ has the highest probability of capture, the student can justify the claim that the plant species has adapted aerodynamically to its own species of pollen. Within the files of MODEL there are a number of such adapted species, as well as a number of species that are not adapted to the aerodynamics of their own pollen–species. Through trial–and–error, the student gradually learns the nature of the physical principles that both influence pollen–cap– ture and dictate efficient or inefficient wind pollination.

THE LECTURE AND CLASSROOM FORMAT

Computers are used only after students have been given information through a lecture format. The biological concepts that underlie TREE and MODEL are given as an extension of lecture material that would normally be presented in a standard botany course. Thus, the communication of information is not stilted toward the use of computers, but rather it is based on the pedagogical principles of botany.

Within the classroom environment the student is introduced to the mechanics of using TREE and MODEL, e.g., how to call up files; how to enter the variables that influence the outcome of the simulation; how to get hard copies. The student is then given a problem to solve that will open an area of plant biology to quantitative analysis, e.g., what is the "best" tree for light-interception in the tropics? What is the "best" pollen type for a pine tree? The computer is then used as a tool to answer questions. To help the student, biological materials are displayed around the laboratory setting, e.g, models and photographs of trees, microscope preparations, pine pollen. In addition a laboratory manual, containing suggested ways of solving the problem is distributed. The manual places the problem in context and contains factual information relevant to the plants involved in the exercise; trial simulations with expected results (so the student is made confident that the programs are being used correctly); and a follow-up discussion of how to evaluate data and assess their relevancy. The computer is used as a tutorial tool, as well as an experimental tool. Rather than giving "the answer" to the student, the computer is used to test hypotheses and examine their consequences. Our objective is to help students learn how to phrase and answer questions that, as yet, do not have clear answers. There may be only "one answer" to a given question, or there may be "many answers" which indicate that the question is subtle with regards to its biological implications. In this environment, both the student and the instructor operate as scientists seeking a solution rather than as the receiver and sender of clear-cut answers.

PROGRAM CAPABILITIES (SUMMARY)

TREE can:

a. Compute the mechanical stress on any specified branching pattern.

b. Compute the light–interception of any pattern, correct for self–shading, and predict the shadow that will be cast on nearly branching patterns.

c. Determine the ratio of a to b to quantify the reconciliation between these two features of plant growth.

d. Calculate a–c as a function of changing latitude.

e. Compute a "universe" of all possible branching patterns defined by the convariance of various geometric parameters that define shape.

f. Calculate the most parsimonious trajectory of c which would improve performance in e.

g. Predict the morphological changes (that would maximize c in e) through plant evolution.

MODEL can:

a. Quantify the speed and direction of airflow motion around any female plant organ.

b. Quantify the behavior of any pollen grain type in a.

c. Predict the optimal size and shape of pollen grains that would maximize pollination.

d. Provide a complete spectrum of pollination efficiency for all possible pollen grains moving within a.

e. Be used to relate the predicted "optimal" form of pollen to the actual physical properties of the pollen from a plant species.

f. Predicts the relationship between a female plant organ's form and function (as a pollination device).

CLASSROOM ENVIRONMENT

a. The computer is used as an experimental tool in a laboratory/classroom environment.

b. Lecture format provides the background for conceptual issues, the principles of which are discussed from a strictly botanical perspective.

c. In the laboratory, students are given a brief description of the way TREE and MODEL and be used to answer biological questions.

d. This is followed by the posing of a biological question which can be addressed by means of the computer.

e. Supplemental information (photographs, microscope demonstrations, living material, etc.) is presented in the laboratory, and a manual is distributed which

f. Shows the student how the computer can be used to solve numerical aspects of the question asked.

g. The student explores the problem by experimenting with computer-based predictions and relating these predictions to real data.

h. The student and instructor enter into a dialogue about the question's possible answers and their biological significance.

REFERENCES

Niklas, K. J. 1986. Computer-simulated plant evolution. "Scientific American" 254(3):78-86. _____ and V. Kerchner. 1984. Mechanical and photosynthetic constraints on the evolution of plant shape. "Paleobiology" 10:79-101.

CRYSTAL

Presenter: Ed Mulligan *University of Washington*

Crystallography is a long studied but important topic in Geology. Much of a geologists time is spent studying minerals that under ideal conditions, will form geometric shapes of classic beauty. The geometry of crystals is so well defined,that some professors describe crystallography as one of the few closed topics in science. The problem with crystallography is due to the nomenclature, which often strikes the new student as arbitrary and illogical. Imagine trying to learn a coordinate system that may have three or four coordinate axis, the axis are not necessarily perpendicular, the units of length vary according to the axis, and the distances are inverted and cleared of fractions before use. Current methods of teaching include lectures, manual stereonet plots, and little wooden blocks.Personal experience has found these methods to be deficient.

Crystal symmetry is the general name for a repeating pattern of crystal faces that indicates the ordered internal arrangement of the substance. There are many different kinds of symmetry,but we shall consider only the symmetry found in simple convex crystals. All crystal symmetry properties can be expressed as a combination of rotation and inversion. Given the symmetry of a crystal, specifying a crystal face automatically gives additional related faces. In a highly symmetric crystal (i.e. isometric),naming a single crystal face will generate the complete shape.

All possible crystal shapes are found in the seven crystal systems. Each of the seven systems has an identifying symmetry combination. The seven crystal systems are subdivided into a total of thirty–two point groups. Each point group is defined by a different minimal set of symmetry operators. Choosing a point group completely defines the symmetry operators the crystal must have. The point group with the highest order of symmetry is called the hexoctahedral class and has the symmetry operators"4/m 3 2/m". Seven crystal forms are found in the hexoctahedral class, ranging from the cube with six faces to the hexoctahedron with forty–eight faces, depending on the orientation of the single face entered to describe the form. Every other point group has multiple forms also, all depending on the orientationof the key crystal face generating the form.

The orientation of crystal faces is described by a notation called Miller Indices. The indices are calculated by measuring the distances (in the respective axial units) from the origin to the crystal axis intercepts of the face, inverting the distances,and then clearing the numbers of fractions. Keep in mind that crystal axes do not have to be perpendicular, nor do the axial unit lengths have to be equal. As an example, the Miller Index,(012), means that the specified crystal face is parallel to the"a" crystal axis, intersects the "b" axis at 1 "b axial unit",and intersects the "c" axis at 1/2 "c axial units".

A stereonet (Wulff Equi-Angular Net) is a 2-dimensional graphical aid for interpreting some 3-dimensional objects. In the case of the program "CRYSTAL", the stereonet aids the user in determining the orientation of the crystal faces, and in finding the crystal symmetry elements. To understand a stereonet, assume a sphere encloses the crystal. Extend vectors normal to each crystal face from the center out to the sphere. Project each point of intersection with the sphere onto a horizontal plane bisecting the sphere such that the projection lines pass thru the North or South Pole of the opposing hemisphere. Using the angled projection as described above will retain angular relationships and reduce the edge distortion of a vertical projection. Each crystal face will have a point plotted on the stereonet circle implying a direction for the crystal face normal vector. Crystal symmetries are visibly illustrated on the stereonet as repeating patterns of points.

One of the goals for the Geology portion of the IBM Olympus project is to produce a program that will allow a student to interactively create a crystal by choosing the symmetry and the orientation of the key faces. Once the crystal is generated,the student should be able to rotate the crystal to any view and simultaneously have a stereonet of the view showing the symmetry elements for that view. The program should help the student learn crystal symmetry, the crystal systems, the point groups, Miller Indices, and the interpretation of stereonets.

The "CRYSTAL" program, created by the Dept of Geological Sciences at the University of Washington under the auspices of the IBM Olympus Grant, provides the user with the ability to create and display crystal shapes. The "CRYSTAL" program runs on an IBM PC, PC/XT, or PC AT equipped with color graphics. The program was written in standard Fortran-77. It is compiled with the IBM (Microsoft) Fortran compiler. The program is presently over 4100 lines of code, not including the graphics subroutines.The graphics are provided by calls to a library written by the UW Olympus Project. The graphics library will support pen plotters.

The user of the program, "CRYSTAL", selects the crystal system and the symmetry point group from two levels of menus,and then enters the orientation of the required crystal faces.The faces can be entered via Miller Indices or by using plane equations. When using Miller Indices, a distance factor will be requested since Miller Indices specify only an orientation. By careful manipulation of the distance factor, the user controls the actual shape of the crystal. Once the user has entered the crystal definition, the program will then generate the crystal shape. Since generation of complicated crystals can take up to ten minutes and several tries needed to get the face parameters entered correctly, there is an option to store and recall crystal definitions. The crystal is displayed on the left side of the screen with hidden lines in a different color. The original view of the crystal shape is looking down the "c" crystal axis, the"b" axis is found in the horizontal plane, and the "a" axis falls where it will as determined by the entered mineral data. The right half of the screen displays a stereonet of the crystal,which helps with the 3D interpretation by the student. By using the arrow keys on the numeric keypad, the crystal can then be rotated to any view, in real time, with the corresponding update of the stereonet clearly showing the symmetry elements of the crystal. A graphics command menu allows various options such as saving the image, executing a command file, or using a plotter.

The program works by converting the Miller Indices and distance from the center for each key face into a plane equation of the form AX + BY + CZ = D where D is positive. The list ofcrystal face plane equations is then processed by a subroutine that applies the symmetry operators of the chosen point group to each equation in the list. All new linearly independent plane equations are added to the bottom of the list. Once the list of crystal face plane equations is complete, the program will then calculate the crystal corner points, and then figure out which point belongs to which face and in what order. The method for generating the crystals is similar to a program called "SHAPE" byEric Dowty (American Mineralogist, Vol 65, p 465, 1980). A new list of corner points and vectors normal to the crystal faces is created by reorganizing the crystal data. This data structure is primarily a list of (X,Y,Z) coordinates and thereby allows a fast crystal rotation by use of matrix multiplication. The crystals are drawn hidden faces first in one color, and then the visible faces in another color. The decision on whether a face is hidden or visible depends on the sign of the Z coordinate of the vector normal to the crystal face. The stereonet is figured by using a mathematical algorithm for the projection applied to the crystal face normal vectors stored in the new list. The hidden crystal faces are plotted first and then the visible faces.

A special option available to advanced geology students is the capability of generating optical interference figures in 16 colors on the same screen with the crystal shape and stereonet.This option will only work on IBM PC compatible microcomputers equipped with an enhanced graphics adaptor and matching monitor. Interference figures are used to help identify mineral grains found in rock thin section mounts by using the petrographic microscope to view the thin sections. The crystal orientation in relation to the refraction indicatrix has subtle but profound effects on the interference figure produced in a petrographic microscope, an important concept for the more advanced geologist,but often times difficult to demonstrate as one rarely finds a mineral grain with the proper orientation to illustrate the point being made. The theory behind interference figures is quite complex, so I will not delve into the details, but the basic idea is that polarized light can be used to analyze the characteristic optical properties of a mineral. The important part is that the figure can be mathematically predicted given the mineral data.The colors found in an interference figure tend toward washed out pastels, but careful selection from the EGA color range has produced a palette that approximates 16 colors found in the first 3 orders of interference colors. The original idea for the interference figure option came from a program by Craig Bethke (American Mineralogist, Vol 65, p 1294, 1980) but were placed his numerical solution code with an algorithm by Jim Nicholls (Univ of Alberta, unpublished). Mark Ghiorso (Univ ofWash) put much effort into the interference figures and was rewarded with a speed increase on the order of 2 magnitudes, but an AT still needs three minutes to calculate an interference figure. We plan to adapt the program to drive an AT equipped with an IBM Professional Graphics Adaptor and Monitor to produce figures with more realistic colors in the near future.

The "CRYSTAL" program is in the final stages of development so it has not been used in the class room setting by students,but all of the professors and graduate students that have seen it are excited about the impact the program will have in teaching mineralogy and optical petrology. The program will not replace wooden blocks, but when used in conjunction with the existing teaching aids, the important concepts are illustrated much more clearly. A good example is the three-fold roto inversion axis on the hexoctahedron. Most students have a difficult time when looking for this symmetry axis, but it is clearly shown by the program. The possibilities go on and on.

The program "CRYSTAL" is the first step in some ambitious plans by the Geology Department to use more computer technology as an aid to instruction. Most oil and mineral companies make extensive use of computers for everything from remote sensing to mathematical modeling. Thanks to the IBM Olympus Project at the University of Washington, the Geology Department is installing a micro-computer graphics lab consisting of 10 workstations linked to a Token Ring Local Area Network. Graphics output devices on the network will include a large pen plotter for drawing maps, a small pen plotter for various geologic diagrams, and two color raster devices for dumping screen images. Our students will benefit by using these new instructional resources to aid their education. Thank You IBM.

DYNAMIC MODELS IN BIOCHEMISTRY: SIMULATIONS USING SPREADSHEET SOFTWARE FOR MICROCOMPUTERS

Presenter: Sandra I. Lamb *University of California, Los Angeles*

Coauthors: Sandra I. Lamb *University of California, Los Angeles*
Daniel E. Atkinson *University of California, Los Angeles*
Steven G. Clarke *University of California, Los Angeles*
Douglas C. Rees *University of California, Los Angeles*
David S. Barkley *University of California, Los Angeles*

There are a number of types of quantitative problems that appear in biochemistry textbooks that can be solved readily using almost universally available hand-held calculators. These problems can also be solved using sophisticated computers. The use of modern computer systems makes it possible to expand such problems to allow students to see the effects of changing several variables and plotting results. Of course, to do this, students must have both access to a computer and enough knowledge of programming to set up the problem. Since this is not the case for many undergraduate students, an alternative approach has been taken which utilizes microcomputers with electronic spreadsheet software including graphing capabilities. This software (typified by Visicalc, Supercalc, Lotus-1-2-3, Lotus Jazz, Microsoft Multiplan and Excel) was originally designed for financial-accounting purposes but we have found it to be extremely useful for biochemistry and chemical calculations in general.

The reasons for using such an approach include the increasing availability of microcomputers to students (and the increasing knowledge of how to use them) and the simplicity of programming equations into spreadsheet software. Moreover, modern spreadsheet and graphing software, such as Supercalc3, version 2.0, not only has most scientific notation, but includes the capability of graphing any equation that may be expressed mathematically. We have found that, even without any programming experience, students can begin working problems within several hours of their first computer interaction. With the graphics capabilities of the newer spreadsheet software, it is also possible that students can immediately visualize the complex functions that come out of many chemistry and biochemistry problems.

Many kinds of relationships in biochemistry can be made much more understandable and intuitive when the student can work with a model or simulation that allows him/her to vary conditions, or the properties of such components as enzymes and then observe the results. These properties underlie the interactions between sequences of chemical reactions that maintain homeostasis in living systems. Many simultaneous equations would have to be solved to explain these interdependent conditions. Using the spreadsheet software allows the student to set up the conditions, with the solutions being carried out by the program before the students' eyes. Then the visualization of results by viewing a graph gives immediate gratification in terms of understanding relationships.

When pre-programmed or "canned" models are used, the student's interaction is relatively passive, and the model is little more than a black box that magically churns out answers. The critical relationships tend to be hidden. Spreadsheet programs, although originally designed for use in business, are ideal for simulating variable interactions in chemistry and biochemistry. This is because relationships are the very basis of the spreadsheet approach. Instead of using a canned program, the student can set up the model seeing at every step the underlying trends. Because the model is fully exposed at all times, modifications and extensions are easily incorporated.

The obvious clarity of the model (as opposed to the interpretation of say, Fortran code), is an inherent pedagogocal advantage. Making difficult problems into simple calculations and then plotting the results of those calculations immediately brings the relevance of the model system to the student simply and immediately (see Randall Davis, Knowledge Based Systems, Science, 231, 957 (1986)).

We have prepared a ten-page introduction to the use of our PC-Networked XT student systems as well as a 30-page introduction to the use of Supercalc3 version 2.0 (now available in our PC Lab in brilliant color on IBM Enhanced Graphic Displays). After working several simple problems, the student is asked to work through several examples from the forthcoming textbook: Dynamic Models in Biochemistry: A Workbook of Computer Simulations using Electronic Spreadsheets (Daniel E. Atkinson, Steven G. Clarke, Douglas C. Rees and David S. Barkley, Ed.)

In all of the exercises in this book, personal computers are used as tools for simulation devices to build and interrogate models of complex events and ideas in disciplines where such simulations have enjoyed only modest past successes. One possible approach to the solution of complex chemical and biochemical interactions is to provide libraries of special purpose programs with extensive graphics displays for realistic simulations of chemical and biochemical events. Not only are such programs very expensive and often difficult to learn, they preclude the student's actual involvement in building his own modeling system.

We believe that is will be several years before most undergraduate students will come to the classroom with even rudimentary programming skills. Even when this day arrives, probably only a few students will have the talent or desire to spend the hours and days required to build software models of complex events using procedural languages such as BASIC, Pascal or C. However, the day is already here when students can build sophisticated computer simulations by using non-procedural languages. For the moment, we believe that the best choice of an easy-to-learn, non-procedural computer language is the electronic spreadsheet that includes graphing capabilities.

Problems in biochemistry, involving extensive model building in the fields of enzyme kinetics, metabolism and its regulation, membrane transport, and the structure and stability of macromolecules, are included in the aforementioned text. The Instructor may assign problems in any given topic in sequence with the class discussion, or the student may be allowed to work through the problems at his/her own rate. We have found that students do require an assignment when they can accomplish the work in the quickly rewarding environment of the Undergraduate IBM PC Laboratory in the Chemistry Department of UCLA.

We are presently working on and will introduce similar problem solving techniques in our introductory Physical Chemistry course. An additional benefit from this approach has been the involvement of faculty members in the use of spreadsheet programs for more advanced chemical instruction. Some of these faculty might not be interested in the use of computers without such an inducement.

INTELLIGENT CAI

Presenter: Martin Gardy *Cornell University*

Coauthors: Martin Gardy *Cornell University*
W. D. Hagamen *Cornell University*

Let us begin by outlining some of the features that a computer program should have in order to be an effective teacher.

1. It has to be able to think and have the ability to explain its reasoning. Teachers are not selected because they are the repositories of facts which may be extracted if one phrases the question correctly. They must be able to reason and teach their students to reason.

2. It must be a good listener. This assumes not only that it understands what the student is saying, but must do this under adverse conditions where students make typographical errors and use ungrammatical sentences. It also means that the program does not focus exclusively on whether an answer is right or wrong, but can evaluate the logic of a student's reasoning.

3. It needs to know how to teach without being told. Authors in conventional computer mediated tutorials need to provide detailed program instructions [1,2]. These include a list of questions and a series of anticipated answers for each, with varying degrees of correctness. For each anticipated answer, the author must decide what the program should say and then what should be asked next. This must also be done for student responses that do not match any of the anticipated answers. Not only is such authoring a time–consuming process, but it is not what teachers are trained to do. As a result, the best teachers usually do not get involved in CAI. This situation would be reversed if the teacher could give the program some basic facts and the program could take it from there.

This paper will describe two entirely separate programs used to teach medical students. In a sense, internal medicine and gross anatomy represent opposite extremes within the spectrum of medical education. Medical diagnosis (MEDCAT) [3] is heavily dependent of deductive reasoning. Gross anatomy (CATS) [4] is a descriptive discipline which utilizes inductive reasoning to reduce the need for memorization by organizing things into logical groups.

In this brief paper we will describe some of the things the programs will do. We will not explain the mechanisms by which these are achieved. For such information the interested reader should refer to a more complete description [5]. Both programs are coded in APL and implemented on the IBM PC/AT.

CATS (Computerized Anatomical Teaching System)

It is important to realize that there are no prestored answers or questions in either MEDCAT or CATS. Rather, the programs figure out the answers. The necessary logic is stored as a semantic net – a three row integer matrix indicating the relationships between the concepts within its domain of knowledge. In CATS the pointers indicate the nature of the relationship, while in MEDCAT they primarily represent the strength of the relationship. The programs also do not use any key words. Syntactic markers (prepositions, conjunctions, punctuation, etc.) are used to parse the input. The names of things are stored as a list of noun phrases, including synonyms.

CATS is subdivided into five programs or workspaces – one for each of the five major regions of the body (Abdomen & Pelvis, Upper Limb, Lower Limb, Thorax, and Head & Neck). For each of these regions, the subject matter expert entered approximately 300 noun phrases (names of things) and about 1000 pointers (logical relations). This data could be displayed on about three pages. On the other hand, if asked to print out all it knows about a region, it will type over 300 pages of narrative text. This is equal to (actually exceeds) the amount of material in a standard unabridged textbook. We tell the students that we can never ask them anything on the written exams that the program cannot answer. From the viewpoint of the subject matter expert, this represents a tremendous saving and is the reverse of the situation encountered in conventional CAI. In a pedagogical sense, it represents the distinction between learning by rote and using reasoning to decrease the amount of memorization.

CATS normally takes the initiative in asking questions of the student. However, students may interrupt at any point and ask questions of their own. We shall illustrate the latter mode because it will permit us to demonstrate the power of the program to understand English and to reason from a small knowledge base.

In the following examples, what the computer prints is shown in upper case, while what the student types is shown in lower case. Actually, all interactions are carried on in upper case, which is the normal APL character set.

CONTEXTUAL UNDERSTANDING

1. What is the innervation of the biceps?
THE MUSCULOCUTANEOUS NERVE.

What is its action?
IT FLEXES THE SHOULDER AND ELBOW AND SUPINATES THE FOREARM.

On the simplest level, the pronoun reference seen here is achieved by storing (the node numbers for) the subject and object (answer) of the previous question. The more interesting question is how the program knows which the pronoun should refer to – the BICEPS or the MUSCULOCUTANEOUS NERVE? The answer is that in the program's experience muscles, not nerves, have actions. In this regard, consider the following sequence.

What is the innervation of the biceps?
THE MUSCULOCUTANEOUS NERVE.

What else does it innervate?
THE SKIN OF THE LATERAL FOREARM, THE BRACHIALIS AND CORACOBRACHIALIS.

Here it chose the nerve because in its experience nerves, not muscles, innervate things.

2. What anastomoses would develop following an occlusion between the second and third parts of the axillary?
AN OCCLUSION BETWEEN THE SECOND PART OF THE AXILLARY ARTERY AND THE THIRD PART OF THE AXILLARY ARTERY WOULD PRODUCE THE FOLLOWING ANASTOMOSES:

THE DEEP BRANCH OF THE TRANSVERSE CERVICAL ARTERY TO THE LATISSIMUS DORSI TO THE THORACODORSAL ARTERY.

THE DEEP BRANCH OF THE TRANSVERSE CERVICAL ARTERY TO THE SUBSCAPULARIS TO THE SUBSCAPULAR ARTERY.

THE LATERAL THORACIC ARTERY TO THE SUBSCAPULARIS TO THE SUBSCAPULAR ARTERY.

THE SUPERFICIAL BRANCH OF THE TRANSVERSE CERVICAL ARTERY TO THE INFRASPINATUS TO THE CIRCUMFLEX SCAPULAR ARTERY.

THE SUBSCAPULAR ARTERY TO THE INFRASPINATUS TO THE CIRCUMFLEX SCAPULAR ARTERY.

THE THORACO-ACROMIAL ARTERY TO THE ANTERIOR PART OF THE DELTOID TO THE ANTERIOR HUMERAL CIRCUMFLEX ARTERY.

THE THORACO-ACROMIAL ARTERY TO THE ANTERIOR PART OF THE DELTOID TO THE POSTERIOR HUMERAL CIRCUMFLEX ARTERY.

THE THORACO-ACROMIAL ARTERY TO THE LATERAL PART OF THE DELTOID TO THE ASCENDING BRANCH OF THE DEEP BRACHIAL ARTERY.

THE THORACO-ACROMIAL ARTERY TO THE LATERAL PART OF THE DELTOID TO THE POSTERIOR HUMERAL CIRCUMFLEX ARTERY.

Disregard the complicated answer for the moment, and concentrate on the student's question and the program's paraphrasing of the question. Clearly the program extracted more information than was explicitly given. How did it know, for example, that AXILLARY refers to the AXILLARY ARTERY, rather than the AXILLARY NERVE or AXILLARY VEIN? It does this the same way people would. From its experience (semantic net) it can determine that only the axillary artery, not the nerve or vein, is subdivided into named parts, and that only it can be involved in anastomoses.

How does it know that SECOND uniquely defines the SECOND PART OF THE AXILLARY ARTERY, rather than the SECOND RIB or SECOND DORSAL INTEROSSEOUS, within the context of the discussion? It uses the same logic already described. The second rib, second dorsal interosseous muscle, and other structures containing the word SECOND could not take part in arterial anastomoses.

3. What is the innervation of the second and third lumbricals?
THE MEDIAN NERVE.

People would have no difficulty in knowing that SECOND, in this context, refers to SECOND LUMBRICAL. Neither does the program. However, it cannot use the reasoning described above, because most of the other structures containing the word SECOND (arteries, bones, skin, as well as muscle) all are innervated by nerves. The program does exactly what we would do – it "borrows" the word LUMBRICALS from the second phrase and appends it to the first.

4. What are the relations of the wrist?
IT IS POSTERIOR TO THE RADIAL AND ULNAR ARTERIES, THE MEDIAN AND ULNAR NERVES AND THE TENDONS OF THE FLEXOR CARPI RADIALIS, FLEXOR CARPI ULNARIS, FLEXOR DIGITORUM PROFUNDUS, FLEXOR POLLICIS LONGUS AND PALMARIS LONGUS. IT IS ANTERIOR TO THE TENDONS OF THE EXTENSOR CARPI RADIALIS BREVIS, EXTENSOR CARPI RADIALIS LONGUS, EXTENSOR CARPI ULNARIS, EXTENSOR DIGITI MINIMI, EXTENSOR DIGITORUM, EXTENSOR INDICIS, EXTENSOR POLLICIS BREVIS AND EXTENSOR POLLICIS LONGUS.

What are the anterior relations of the wrist?
IT IS POSTERIOR TO THE RADIAL AND ULNAR ARTERIES, THE MEDIAN AND ULNAR NERVES AND THE TENDONS OF THE FLEXOR CARPI RADIALIS, FLEXOR CARPI ULNARIS, FLEXOR DIGITORUM PROFUNDUS, FLEXOR POLLICIS LONGUS AND PALMARIS LONGUS.

This illustrates the role of adjectives in restricting the scope of a question. In the first example, the program gives all the structures crossing the wrist; in the second, this is restricted to those passing anterior to it (that the wrist is posterior to). Any logical variation of this produces the expected results. For example, "What are the anterior and posterior relations of the wrist?" produces both answers. Effectively this restricts the meaning of the predicate nominative ("are relations of") to whatever adjectives modify "relations". If there are no modifiers, all relations are listed.

5. What are the actions of the biceps?
IT FLEXES THE SHOULDER AND ELBOW JOINTS AND SUPINATES THE FOREARM.

What are its actions on the forearm?
IT SUPINATES THE FOREARM.

The domain of such verbs may also be limited by prepositional phrases. "What are the actions of the biceps?" produces three verb–object pairings (flex – shoulder, flex – elbow, and supinate – forearm). Here the prepositional phrase eliminated those actions on objects not included in the prepositional phrase ("on the forearm").

6. What is the origin of the extensor acrpi rdialis brevis?
THE LATERAL SUPRACONDYLAR RIDGE.

Here the student misspelled two words (CARPI and RADIALIS), and the program correctly interpreted the meaning. The program knows that there are only two structures that contain the words EXTENSOR and__ BREVIS: EXTENSOR CARPI RADIALIS BREVIS and EXTENSOR POLLICIS BREVIS. Having limited the ambiguity in this way, the misspelled words have only to be compared to the words in these two noun phrases. This is done by means of the closest character match, which is essentially an APL primitive function.

Each of these examples illustrates a different way in which the program can correctly interpret the meaning of a student's question, based on the context of the discussion. These are powerful linguistic tools and permit a freedom and naturalness of expression which is almost essential if a program is to handle free format input.

ALGORITHMIC EXPANSION OF KNOWLEDGE

We began by saying that intelligent CAI programs do not require that all information need be explicitly entered into the database. Instead, the program should be able to deduce more complex relations. The question and answer sequence (2) about collateral arterial pathways given earlier may serve to illustrate this.

To answer this question the program must go through the following steps. (1) It finds all the arteries proximal to the occlusion (first and second parts of the axillary and the subclavian artery). (2) It then determines all the direct branches of these vessels. (3) It then traces through the matrix again to find the branches of these branches until it can determine all the muscles they supply, since collateral circulation develops only in muscle. (4) It then determines what other arteries supply these same muscles. (5) It traces each of these back up to the artery distal to the occlusion. (6) Lastly, it generates a clear narrative description. It does all this with a reasonable response time – about three seconds.

The decision to handle complex questions in this manner involved a choice. Clearly, all the muscles supplied by each artery could have been directly represented in the matrix, as could all potential collateral arterial pathways. The decision to have the program use reasoning to determine the answer each time was based on the following considerations. (1) To do otherwise would greatly increase the size of the semantic net. (2) More importantly, it represents the distinction between learning by rote and using reasoning to decrease the amount of memorization. No experienced anatomist memorizes pathways for collateral circulation – they figure them out when the need arises. Getting students to use the same approach and to develop the same skills is a primary task of good instruction.

DISCOVERING GENERAL PRINCIPALS

Gross anatomy is a descriptive science. Unlike medical diagnosis, it does not normally deal with cause and effect. Nevertheless, we encourage students to ask "Why is ..... true?". CATS is programmed to answer this type of question, as we do, by trying to point out general principles. The purpose is to further reduce the need for memorization.

7. Why does the pronator teres flex the elbow?
ALL MUSCLES THAT ORIGINATE FROM THE MEDIAL EPICONDYLE OF THE HUMERUS FLEX THE ELBOW JOINT.

ALL MUSCLES THAT PASS ANTERIOR TO THE ELBOW JOINT FLEX IT.

8. What is the innervation of the first dorsal interosseous?
THE DEEP VOLAR BRANCH OF THE ULNAR NERVE.

Why?
ALL MUSCLES THAT ARE PART OF THE HAND ARE INNERVATED BY

THE DEEP VOLAR BRANCH OF THE ULNAR NERVE, EXCEPT THE THENAR GROUP, 1/2 THE LUMBRICALS, AND THE PALMARIS BREVIS.

ALL MUSCLES THAT INSERT ON THE LATERAL BAND ARE INNERVATED BY THE DEEP VOLAR BRANCH OF THE ULNAR NERVE, EXCEPT 1/2 THE LUMBRICALS.

ALL MUSCLES THAT PASS ANTERIOR TO THE METACARPOPHALANGEAL JOINTS ARE INNERVATED BY THE DEEP VOLAR BRANCH OF THE ULNAR NERVE, EXCEPT THE FLEXOR DIGITORUM PROFUNDUS, FLEXOR DIGITORUM SUPERFICIALIS, AND 1/2 THE LUMBRICALS.

ALL MUSCLES THAT EXTEND THE INTERPHALANGEAL JOINTS ARE INNERVATED BY THE DEEP VOLAR BRANCH OF THE ULNAR NERVE, EXCEPT 1/2 THE LUMBRICALS.

In order to answer why (7), the program focuses on the answer to the previous question. It first searches the matrix for the verb and object used. In this example, using "flex" and "elbow", it would come up with all the muscles that flex the elbow. It then uses these node numbers (muscles) to search again, one relation (verb) at a time, to determine the greatest generalization it can make. When it compares the muscles that arise from the medial epicondyle with those that flex the elbow, it finds a 100 percent correspondence and applies the "all" to the answer. If it cannot make a meaningful generalization by focusing on the verb and object, it will try the subject and verb (8). Here it searched for "deep volar branch of the ulnar nerve" and "innervates" to find all the muscles innervated by the nerve. If it cannot find a complete correlation, it will look for "all but" where the first assertion is more than 50 percent true, and the exceptions are three or less in number. In order to state the generalization more concisely, and to reduce the number of exceptions to meet the requirements of the algorithm, it tries to place the exceptions into groups. In the second example, six muscles become three exceptions by being expressed as "the thenar group" (3 muscles), "1/2 the lumbricals" (2 muscles), and "the palmaris brevis".

Such explanations may be requested by the student, as seen here. They also may be volunteered by the program whenever the student answers a question incorrectly.

The extent of the ability of CATS to discover general principles is illustrated by the following data. For the upper limb, the program generated 846 unique questions. When asked "Why?" for the answers to each of these questions, it was able to give explanations for 677 (80 percent). The total number of explanations given was 1668, or an average of 2.46 explanations per question. Of these 1668 explanations, 308 were unique. Such redundancy is what one should expect from the algorithm. An explanation with the general format "ALL THE ..... THAT ..... THE ..... ARE ....." would apply equally to every member of the set referenced by "ALL".

The attribute that most clearly distinguishes effective teachers in a subject such as anatomy is the ability to reduce the mass of material to a more manageable number of general principles. Acquiring this skill takes years of experience. When we originally started to implement CATS we thought of it as a very efficient method of storing gross anatomical knowledge and producing drill and practice sessions without the laborious authoring requirements of conventional CAI programs. Its ability to generate meaningful general principles to explain the answers to questions, and to do this with an average response time of less than 3 seconds, was a very real and pleasant surprise. It is a feature that greatly enhances the teaching ability of the program.

We have stated that gross anatomy is a descriptive science; it also is a visual discipline. Students are expected to be able to draw the structures that comprise the human body. At CUMC we have largely replaced the textbook with our own illustrated handbook. This handbook, together with the colored drawings, is being put on the computer. It is assumed that students will use this graphic version before signing onto CATS. Eventually the two programs will be combined so the answers that CATS gives are accompanied by detailed anatomical illustrations generated on the screen. These drawings are displayed via the Enhanced Graphics Adaptor (IBM PC/AT). It may be of interest that while the data for CATS can be entered fairly quickly (days), development of the graphic pictures is very labor intensive (many months for a major region of the body).

MEDCAT (Medical Diagnosis, Teaching And Consultation)

MEDCAT makes diagnoses from empiric data stored in patient records and explains its reasoning in response to questions (consultant mode). It also uses its logical and communicative skills to instruct medical students in the proper approach to medical diagnosis (student mode). MEDCAT's reasoning can be modified by free format discussion with physicians (learning mode), which will not be discussed in this paper.

CONSULTANT MODE

WHAT IS YOUR QUESTION?
What was your diagnosis?
ALCOHOLIC HEPATITIS AND ALCOHOLIC CIRRHOSIS COMPLICATED BY ASCITES AND HEPATIC ENCEPHALOPATHY.

WHAT IS YOUR QUESTION?
What signs suggested ascites?
THE PRESENCE OF UMBILICAL HERNIA AND PROTUBERANT ABDOMEN ARE CONSISTENT WITH AND THE PRESENCE OF BULGING FLANKS, DULLNESS OF FLANKS, FLUID WAVE AND SHIFTING DULLNESS STRONGLY SUGGEST ASCITES.

WHAT IS YOUR QUESTION?
Why didn't you consider cholestasis?
BECAUSE THE SLIGHTLY ELEVATED ALKALINE PHOSPHATASE, THE ABSENCE OF PRURITUS, THE DEPRESSED SERUM ALBUMIN AND THE ELEVATED SERUM GLOBULIN DO NOT SUPPORT CHOLESTASIS.

WHAT IS YOUR QUESTION?
What tests would make you think of it?
THE MODERATELY ELEVATED GGTP AND THE ELEVATED SERUM CHOLESTEROL WOULD BE CONSISTENT WITH, THE MODERATELY ELEVATED ALKALINE PHOSPHATASE AND THE SLIGHTLY ELEVATED SGOT AND SGPT WOULD STRONGLY SUGGEST CHOLESTASIS.

As illustrated, MEDCAT can make multiple diagnoses, describe the relationships between diagnoses and also indicate the relative importance of each piece of evidence used in arriving at its conclusions ("consistent with" or "strongly suggest"). It sounds like a physician because the program incorporates diagnostic strategies commonly used by physicians.

The simplest is the pattern-match recognition of a disease - "the typical picture". When this is not helpful, the diagnostician employs a hierarchical approach which groups diseases into increasingly specific categories. For example, the viral hepatitides grouping consists of Hepatitis-A, Hepatitis-B, Hepatitis non-A non-B, Infectious Mononucleosis, CMV, Mumps, etc. This serves a number of purposes. The physician is not burdened by having to think of many specific diseases at any one time. The clinical characteristics associated with the group can be inquired about or looked for to see if this group of diseases is worthwhile pursuing in more detail. The mechanisms for grouping are quite varied, but usually involve setting up competing or dichotomous categories. Support for one line of reasoning may mean the weakening of the alternative. For example, groupings can be done by anatomical localization (small bowel versus large bowel obstruction), physiology (hepatocellular versus cholestatic jaundice), etiology (infectious versus malignant), etc.

The program also enables us to deal with a disconcerting paradox. We teach medical students the virtues of thoroughness when, in reality, the hallmark of the expert diagnostician is incisiveness. Problems are rarely solved by non-selective consideration of all possibilities. The justification for performing multiple tests and procedures on "educational grounds" is no longer feasible nor tenable. The efficiency of the expert's search stems partially from the ability to recognize hierarchical groups, but also from the more versatile use of the evidence. Students instinctively use the presence of findings to suggest diagnostic possibilities, but have difficulty in using the presence of a finding or abnormal result to reject possibilities. They have even more difficulty in using the absence of findings (presence of normal results) to support or argue against the possibilities being considered. The program makes these strategies very apparent and offers the student the opportunity to be thorough, not as an end, but as a means towards becoming more selective.

STUDENT MODE

When MEDCAT is used to help medical students sharpen their diagnostic skills, the program plays the dual roles of (1) providing patient data as requested by the student and (2) functioning as a tutor to evaluate and guide the student's reasoning.

The student selects a patient and, after the diagnostic subroutine is executed, is given a brief narrative summary which includes the patient's demographic data, chief complaint and a few of the initial presenting symptoms. The student is then instructed to ask for any empiric patient data needed to make the diagnosis. This is an interactive process. The program may interrupt and ask the student the types of questions that we believe a "good" tutor uses.

These strategies depend on the fact that the program knows (1) which steps in the reasoning are essential to the diagnosis, (2) which additional ones should be pursued even though they will eventually prove to be false leads and (3) which ones do not make any sense. It also knows at any point during the conversation where the student stands – (1) what factual information he has obtained, (2) what conclusions can be drawn from these facts and (3) what additional information is needed to draw further conclusions. MEDCAT uses these two perspectives to direct its questioning of the student.

1. At the beginning, the program encourages the student to request patient data, but also tries to determine why the data were requested and what inferences the student draws from the results.

WHAT IS YOUR QUESTION?
Was there recent contact with jaundiced patients or use of IV narcotics?
CONTACT WITH JAUNDICED PATIENTS WAS ABSENT AND IV NARCOTICS USE WAS PRESENT.

WHAT DOES THE HISTORY OF IV NARCOTICS USE MEAN?
It suggests viral hepatitis.
I AGREE WITH ACUTE VIRAL HEPATITIS.

WHAT IS YOUR QUESTION?
Were scleral or dermal icterus or an enlarged spleen present?
DERMAL ICTERUS AND SCLERAL ICTERUS WERE PRESENT AND SPLENOMEGALY WAS ABSENT.

WHY DID YOU ASK ABOUT SPLENOMEGALY?
I was thinking about portal hypertension
OKAY

2. When the program knows the student has sufficient information to make one or more diagnoses, it tries to determine whether the student can draw the appropriate conclusions, recognize the pertinent evidence that should be used and, when applicable, suggest more specific diagnoses.

DO YOU THINK THE PATIENT HAS CHOLESTATIC JAUNDICE?
No
I DISAGREE. THE PRESENCE OF LIGHT COLORED STOOLS IS CONSISTENT WITH, THE PRESENCE OF CONJUGATED HYPERBILIRUBINEMIA AND JAUNDICE STRONGLY SUGGEST AND THE PRESENCE OF CHOLESTASIS CONFIRMS CHOLESTATIC JAUNDICE. WHAT ADDITIONAL DIAGNOSES DOES CHOLESTATIC JAUNDICE MAKE YOU THINK OF?

3. When the student makes a diagnosis, the program may choose to assess the extent of his understanding by exploring alternative diagnoses.

DO YOU THINK THE PATIENT HAS CONJUGATED HYPERBILIRUBINEMIA?
Yes
I AGREE.
DO YOU THINK THE PATIENT HAS UNCONJUGATED HYPERBILIRUBINEMIA?
No
I AGREE.
WHY SHOULD YOU REJECT UNCONJUGATED HYPERBILIRUBINEMIA?

4. Up to this point, the tutorial has proceeded along "ideal" lines. At times a student may become flustered and may not know what to ask next, or be unable to respond to a question. He may ask for "help" by entering a null response. The program responds in a number of ways. It may provide the correct answers, or it may look at the questions the student has already asked and at the diagnostic implications (pointers) of each. It selects the diagnostic node that has the most student questions directed toward it. It then prompts

the student to draw the correct inference by asking what additional information would be needed to support or rule out that hypothesis.

WHY SHOULD YOU REJECT UNCONJUGATED HYPERBILIRUBINEMIA?
I don't know
THE PRESENCE OF DARK URINE AND URINE BILIRUBIN ARGUE AGAINST AND THE HIGH DIRECT/TOTAL SERUM BILIRUBIN RATIO EXCLUDES UNCONJUGATED HYPERBILIRUBINEMIA.

WHAT IS YOUR QUESTION?
<enter>
WHAT DOES THE PRESENCE OF PRURITUS, MARKEDLY ELEVATED ALKALINE PHOSPHATASE AND TOTAL SERUM BILIRUBIN, AND SLIGHTLY ELEVATED SGOT AND SGPT MAKE YOU THINK OF?
It sounds like cholestatic jaundice.
GOOD.
WHAT ADDITIONAL EVIDENCE WOULD BE CONSISTENT WITH CHOLESTATIC JAUNDICE?
<enter>
THE DARK URINE AND LIGHT COLORED STOOLS WOULD BE CONSISTENT WITH CHOLESTATIC JAUNDICE.
WHAT DIAGNOSES DOES CHOLESTATIC JAUNDICE SUGGEST?

MEDCAT exhibits a tolerance and understanding that befits its role as a teacher. The primary reason for this is that it focuses on the logic of the student's reasoning, rather than the absolute aspects of right or wrong answers that limit so many computer teaching systems. There are many routes to a correct diagnosis, and even paths that ultimately prove to be blind alleys may be worth pursuing – indeed may require investigation. The program can accompany the student along these paths and respond to his reasons, while "patiently" waiting for a point at which the need to look elsewhere becomes obvious. The fact that the program has a more comprehensive perspective does not mean that this must be forced upon its pupil prematurely. The program exhibits intolerance only in the realm of observations (empiric data). When the student fails to obtain essential information, forgets what he has already been told or, most of all, assumes signs or symptoms or laboratory data that were not elicited by his questions, it becomes quite critical.

DISCUSSION

Intelligent CAI has many advantages over the methods that use prestored questions and answers. Not the least of these is ease of "authoring". Providing a program with its knowledge base is what experienced teachers should be good at. On the other hand, formats such as: "If the student says ....., I will say .....; if he or she says ....., I will say ....." are alien to the way people think, and do not make for natural sounding discussions. The teacher is freed of these sorts of considerations if the program "understands" the knowledge domain in which it functions. Such understanding is provided by the semantic net.

Another important feature of intelligent CAI is the ability to understand what is being said based on the context of the discussion, and to resolve ambiguous alternatives by its knowledge of the subject matter. This is much closer to the way people talk than any system of key word analysis can ever be. Such comprehension is possible because of the semantic net. This also makes it possible for the program to generate its own responses, freed of any reliance on prestored answers or answer fragments.

Still another feature – one that we did not describe in this brief paper – is the ability to change the program's knowledge and/or logic by free format discussions. In MEDCAT you may disagree with its reasoning and, if you can convince it of its errors, can win the "debate" and permanently change the program.

The semantic net (pointer matrix) represents both the subject matter (names of things) and the logic (relations between these concepts). The pointers may indicate such diverse things as the strength of a relationship, the nature of the association and reasons for or against decisions made by the system. Not only may this matrix act as the mechanism to perform the reasoning, but these same numbers may be directly translated into the noun phrases and verb phrases which they represent to describe the reasoning. This makes it a model which is very easy to work with for all concerned – the programmer, the subject matter expert and the user.

CONCLUDING REMARKS

We have illustrated some of the linguistic and logical capabilities of the system. The simplicity of the matrix upon which these functions depend suggests that others may want to apply similar techniques to different problems. The fact that the two programs – with very similar code and data structures – can handle two such diverse subjects strongly implies that this general approach should find other educational uses wherever the ability to reason plays an important role. APL happens to be particularly well suited for this type of application and these types of data, but implementation certainly is not language dependent.

ACKNOWLEDGEMENTS

This work was supported in part by grants from the Anonymous Gift Fund of Cornell University Medical College, the Frances L. & Edwin L. Cummings Memorial Fund and the Advanced Education Projects of the IBM Corporation. We would like to thank Mr. Ward Bell for writing certain auxillary processors used in the 8086 version of the programs.

CITED REFERENCES

1. J. C. Weber and W. D. Hagamen, "Medical education – A challenge for natural language analysis, artificial intelligence, and interactive graphics," AFIPS Conference Proceedings, 35, 307–318, November 1969, AFIPS Press, Reston, VA.

2. W. D. Hagamen, D. Linden, M. Leppo, W. Bell and J. C. Weber, "ATS in exposition," Computers in Biology and Medicine 3, No. 3, 205–226 (1973).

3. W. D. Hagamen, M. Gardy, G. Bell, E. Rekosh and S. Zatz, "MEDCAT: An interactive computer program for medical diagnosis," Proceedings NCC'85 54, 111–119, Chicago, IL, July 1985, AFIPS Press, Reston, VA.

4. W. D. Hagamen and M. Gardy, "MEDCAT/CATS: Two contrasting artificial intelligence applications in medical education," Proceedings The Second Conference on Artificial Intelligence Applications, 503–508, Miami Beach, FL, December 1985, IEEE, Los Angeles, CA.

5. W. D. Hagamen and M. Gardy, "The numeric representation of knowledge and logic – two artificial intelligence applications in medical education," IBM Systems Journal 25, 1986 (in press).

INTERACTIVE GRAPHICS FOR DISPLAY AND ANALYSIS OF EARTHQUAKE SEISMOGRAMS

Presenter: Anthony Lomax *University of California, Berkeley*

ABSTRACT

At the Seismographic Station we are developing an interactive graphics package using the IBM Professional Graphics Display to serve as a general tool for displaying and analyzing the ground motions due to seismic waves. The graphics package is designed to improve upon present display methods to give fast, easy access to a multitude of seismic data processing techniques and to integrate with our IBM-PC based digital data acquisition and database systems. The graphics allows the vivid display of seismic wave properties for teaching and is a flexible, easy-to-use tool for research and routine analysis of seismic phenomena.

The design of the graphics package involves careful consideration of color, symbols, program flow and speed of execution. Colors are varied to depict a four-dimensional data set on the two-dimensional screen surface. Symbols are used where possible to prompt the user and to depict display scaling and the current reference frame. The program is interactive and the flow is as unrestricted as possible to give the user complete flexibility in shaping and exploring the data set. In addition, the graphics package is being developed by its users in an active teaching and research environment and so is continuously being tested and modified in response to practical needs.

INTRODUCTION

At the University of California Seismographic Station we are developing interactive software for the IBM personal computer for the display of earthquake seismograms. This work is being done in conjunction with a comprehensive, PC based, seismic data processing system for our Northern California seismograph network.

SEISMOGRAMS

Seismograms are recordings made by a seismograph of the ground motions caused by earthquakes, explosions or other disturbances. These records are used by seismologists to describe the nature of the source disturbance, to study

268

the the physics of wave propagation and to investigate the characteristics of the part of the earth through which these waves have passed.

Historically, seismograms have been recorded on large, paper sheets, on which a fine trace representing ground motions is produced photographically, with ink or by other means. These paper records are valuable for many reasons. Most importantly, they have been in use for many years and are easily recognizable, and consequently have become a standard for presenting seismic data. Also, these records are commonly high resolution analog recordings on which one can make very fine measurements. The paper records, though, have the disadvantages of requiring large storage spaces, of being fixed in scale and processing state, and of requiring photocopying or digitization before being distributed or further processed.

More recently, seismic data have been recorded digitally or converted to digital time series. This gives a much more compact data set than paper records and allows the use of computers to do sophisticated processing and displays. Commonly, digital seismic traces are displayed on a high resolution, vector graphics screen which allows fast updates of many traces with a level of detail similar to paper records. These graphics systems are, however, expensive to develop, and are difficult for a researcher without extensive programming skills to modify for a particular research or educational need.

In the last few years inexpensive microcomputers with medium resolution color raster graphics displays have become widely available. For seismogram display, these new developments allow researchers with moderate programming skills to create low cost, easy to use and relatively powerful display software that can outperform more expensive systems in ease and flexibility of programming and in overall time spent in interactive situations. At the Berkeley Seismographic Station we chose the microcomputers for our seismogram display because of their availability, low cost and ease of use and because we are developing a new digital telemetry system on the same hardware.

PROGRAM DESIGN

The display program is written in IBM PC Professional Fortran using the IBM Graphics Development Toolkit software and is designed for the higher resolution Professional Graphics Display, though the software can drive other

CRT devices. A hardcopy menu option uses the IBM Proprinter and the IBM 7372 six pen color plotter to give paper copies of the displayed seismograms.

The design of this seismic display software involves a few key techniques. First, the program is interactive; program flow is controlled by the user who always has the choice to undo any operation. This gives one the freedom of exploring new ideas and permutations of the basic data with speed and ease without risk of scrambling the data set. A second key design technique centers around creating concise and symbolic displays and user prompts. The user types few strings or numbers, instead single letter codes and the locator are used for most display modifications. Status information is displayed symbolically when possible instead of using text. These elements make the use of the program intuitive and smooth, like a wandering thought process instead of a chore, giving the user the greatest ease in exploring the data. Another important aspect of program design, though invisible to the user, is that the initial input data is never modified. The data set is filtered by various algorithms as it is displayed, but the original data arrays are always left unmodified so that numeric roundoffs do not accumulate.

PROGRAMMING TECHNIQUES

The programming of this seismogram display routine is of interest because it is being done by a seismologist who will be using the software. Because of the programmer's knowledge of the applications of the program, the effectiveness of the software can be constantly tested and judged in the research and teaching environment as it is developed. Many of the display methods and processing techniques are created during this process. This program is also of interest because the code was developed as an outgrowth of a single subroutine instead of being intricately planned beforehand, then programmed. Developing the software from a core routine allows an idea, in this case the display of seismic data, to be created, modified and expanded in an evolutionary manner. This ability for the program to evolve, coupled with the knowledge and ideas of the user–programmer can lead to a more powerful, useful and original program than might be developed by someone not experienced with the goals of the program working with a predetermined software design.

PROGRAMMING RESULTS

One of the more interesting results of the program development process is the use of a "zoom" rectangle to scale the display. A seismic trace, like other observations of nature, is a fractal curve, continuously exhibiting more detail as it is enlarged. Because of this property, and the limited resolution of raster display screens, fast re-scaling of the display is important. Since typing numeric values for a time interval and an amplitude scale is both cumbersome and indirect, an operation was developed where the user specifies two corners of a rectangular subset of the display that is then mapped into the full display area. This whole operation, which is simple to program with the IBM Toolkit software, mimics the first step of the longer thought process one would go through to scale by entering numeric values.

Another unique characteristic of this program is the display of three components of ground motion in three different colors. Because space is three dimensional, the complete description of the ground motions at a point requires recording in three orthogonal directions. With most methods of seismogram display the three traces, when available, are displayed separately, usually with the time axes aligned. With this new display routine the three traces can be aligned this way, but the use of different colors for each component also allows the traces to be enlarged and overlayed to fill the screen, which gives more resolution to each trace and allows more direct comparison between traces. The overlaying of traces simplifies many seismogram analysis techniques, including the rotation of coordinate axes and the comparison of wave pulses on different components.

EDUCATIONAL AND RESEARCH RESULTS

The seismic display program allows researchers and educators at the Seismographic Station to load, display and modify three-component seismic time series in a matter of seconds with minimal keyboard input. This speed and simplicity allows much more frequent use of computer graphics for research and education, since, previously, generating similar displays would have taken considerably more time.

For educational purposes, this software allows rapid and vivid illustration of the principles of seismology and the processes of seismogram analysis. In addition, since the program is interactive and simple to use, a student can learn to use the display and begin to explore a data set in a few minutes. At this point the student can learn by testing ideas, as would a researcher.

It is expected that this software and other graphics routines we are developing will greatly increase our ability to visualize and comprehend the vast quantities of information collected in observing nature. More generally, it is becoming clear that a well designed, interactive graphics system can be a powerful tool, an extension of one's thoughts that greatly transcends the use of computers as specialized number crunchers.

REFERENCES

Bullen, K. E. and Bolt, Bruce A. 1985. An Introduction to the Theory of Seismology. Cambridge, Cambridge University Press.

Foley, J.D. and Van Dam, A. 1982. Fundamentals of Interactive Computer Graphics. Reading, Massachusetts, Addison-Wesley.

Mandelbrot, Benoit B. 1983. The Fractal Geometry of Nature. San Francisco, W. H. Freeman and Co.

IRIS: A TOOL FOR LEARNING DATA ANALYSIS STRATEGIES

Presenter: **Bruce H. Robinson** *University of Washington*

Coauthors: **Bruce H. Robinson** *University of Washington*
Gary F. Anderson *University of Washington*

EDUCATIONAL GOALS

All analytical, physical and freshman chemistry laboratories have the common objective of collecting data (on a chemical system), processing the information according to a formula or an algorithm (e.g. least squares analysis) and displaying (graphing) the resulting information. In addition to getting an "answer" the real goal of the experiment is generally for the student to understand the chemical principles underlying the experiment. To this end, any computer program which assists the student in developing the results should concentrate on enhancing understanding rather than just crunching numbers.

It is important that use of the computer not be an impediment to learning the chemical principles. The process of learning how to use both the computer and the software can often overshadow the process of learning the chemical principle being observed. Therefore, our goal was to provide a very user-friendly package which could not only be easily learned, but could be used in a variety of settings so that the student was not faced with a new program each time a new course was taken or even another experiment was performed. Up to this point, programs in use in our laboratories had focused on the special case (i.e. one package per experiment) rather than providing a general computing tool suitable for analysis of data. Therefore we set out to develop a general program, useful for all courses an not a series of specific programs.

In summary, our educational goals were as follows:

1) Develop a program which would assist in teaching chemical concepts as the necessary data reduction was being performed; but the program will not do the problem.

2) Avoid black box solutions to data reduction where possible.

3) Create a multi-purpose tool which would provide a common interface between the student and the computer in as many laboratories as possible.

4) Develop a program that would require minimal training time for students with no computer background.

COMPUTING STRATEGY:

Part of our computer strategy was defined by the educational goals set out above. On an implementation level, there were other elements of strategy. These include:

1) Menu driven
1) Interactive model definition
2) High visibility of model (4 ways to view work)
3) Unrestricted distribution: no commercial support software
4) Hardware flexibility: runs on CGA, EGA with Mono/Color.
5) Versatile Data interface
6) Graphical and data printout and plotting
7) Professional appearance (assembler routines)
8) Very easy to use but not all-encompassing

Chemical principles and mathematical models can be thought of interchangeably. In general, there is some theoretical model which the student is testing during the process of collecting and analyzing data. For example, there may be a linear relationship between two parameters which can be expressed in the following form:

Parameter_Y = (Factor * Parameter_X) + Constant

The student can create this model by defining four variables, namely P_y, Factor, P_x, and Constant. Factor and Constant will be scalars and P_y and P_x will be vectors. P_x will be generally thought of as the independent variable and P_y will be the dependent variable (or measured quantity). The goal of the IRIS project was to provide software which would allow the students to define and develop the model simply in terms of the parameters of interest. In IRIS, the student creates variables by name, and identifies them as scalar (one value) or vector (multiple values). The model then is built in terms of these variable names. This allows the student to maintain a one-to-one relationship between the software model and the experimental model. The complete model is built by making simple combinations of the basic variables (e.g. a variable

"F*P_x" could be created and defined as the product of Factor and P_x). To this end, a number of math functions such as *, +, −, /, Log, Ln, Sin, etc. are built into IRIS. In addition to providing interactive model definition, IRIS also allows models to be redefined as the need arises. Completed models can be stored to disk.

The above is perhaps a bit cumbersome to those accustomed to using parentheses and nesting operations. However, IRIS is designed to train students to examine the data. By this method then they are forced to proceed stepwise and can examine the results at each step. The positive feature of this approach is that students can find mistakes very quickly. Dynamic redefinition of variables is a feature that permits students to correct mistakes quickly or change relationships.

Once the model is defined, the student also needs to be able to look at the various components. To meet this need for model visibility, IRIS provides a table display of vector variables, a list of variable definitions and a list of scalar values and a spreadsheet interface to the vector data. For instance, the student defined the vectors P_x and P_y. In the spreadsheet mode, the student can look at and modify the individual values of the vectors (visual look of Lotus 123, etc.). The vector data can be either entered manually or read from standard ASCII files on disk. Once in the model, data can be changed, deleted, or suppressed from the processing or graphing.

Least squares is another operation and combined with the "overlay" capability of the graphing menu provides a quick, powerful method to do least squares and visualize the results. Often there is an unknown parameter which cannot be solved for by linear least squares. The student can try different values and examine the results which depend on the values. This is the simplest form of learning to handle non-linear least squares using simple searching and optimizing methods.

Part of the visibility function is of course the graphing. IRIS supports a simple x-y plot format which allows plotting up to three dependent variables against the independent variable. Line color and marker support provide visual differentiation of the plots. All graphics functions such as autoscaling and axis labelling are supported in a transparent manner. The graph is defined on a simple definition screen and is done in terms of the variable names.

Since we are an educational institution and the software is intended to reach many students, we felt the need to be able to provide each student with an individual copy of IRIS. To accomplish this, we decided to make the program completely independent of any commercial product and thus avoid any restrictions based on software license or royalty considerations. Fortunately, we had an extremely capable and competent programmer to accomplish this.

In our department, we have a variety of hardware configuration. In order to maximize the utility of the program, we provide support for the monochrome display adapter, the standard CGA and the EGA in both monochrome and high resolution color mode. Obviously, IRIS will not produce graphics on the MDA. In addition, this flexibility allows students, who have their own IBM equipment or a compatible system, to get maximum use out of the program.

In many experiments, the actual data is taken by means of a data acquisition board connected to the analytical apparatus. Usually this results in much more data than can be typed in manually. IRIS accepts data in a standard ascii format and reads it in from the disc. This has proved to be a very easy way to supply data analysis without generating any new or specific code. Data acquisition programs are chained into IRIS by spawning. The user (student) does not even realize a new program has been brought in. Furthermore, if data acquisition consumes most of the laboratory time, the data may be analyzed later using IRIS as a stand-alone program.

We feel that the visual interface between the computer and the user is an important factor in the learning experience. To this end, we designed the menus, graphics output, and definition screens to have a very professional appearance and work somewhat like packages which are available commercially. We felt that this would not only accomplish the goal of providing user- friendly software, but would prepare students to use packages which they might have available to them in the workplace once they left school.

TECHNICAL IMPLEMENTATION

IRIS is written using Microsoft FORTRAN version 3.3 and depends heavily on a number of utilities written using the IBM Macro Assembler version 2.0. The assembler utilities are used to provide all of the text displays and entry fields for the user to provide information to IRIS. Use of these assembly routines provides much quicker and more professional looking screens than would be possible using standard FORTRAN and an ANSI driver. In addition, all keyboard interfacing is done using assembly language routines. All of the assembly routines access the devices with either the ROM BIOS or standard DOS function calls. We felt this was necessary to insure portability among the different video boards, as well as supporting interfaces on future IBM operating systems and hardware. Some speed was sacrificed due to this tactic. The system is of course quite modular, and parts such as the menu manager are high level FORTRAN calls: other software we develop to have a very similar interfaces. The graphics package is a stand–alone set of library–callable FORTRAN routines.

PROJECT RESULTS:

IRIS functions in such a way that the student is relieved of the drudgery of the calculations but he must ponder the transformations and manipulations necessary to convert the raw data into relationships that can be tested by A (least–squares) model. It is this feature of the program that we find most interesting. The student must actually consider the model and its relation to the initial data. His/her efforts are focused exactly where they should be, and the calculation and graphing aspects are handled by IRIS.

The program has worked very well and is now being integrated into the curriculum of 4 chemistry laboratory courses and 2 chemistry lecture courses. We have found that this program has replaced all previously written special applications programs in our chemistry laboratories. The program is easy to learn and requires around a 1 hour introductory tutorial to explain most of the features. We have found that the approach to data analysis embodied in IRIS has emphasized the intellectual part of the data modelling process which goes on at the research level. Student feedback has been very positive. Students ask for updates of the program after they leave the course and are using the program for other class assignments. Researchers are also using this tool for data analysis since it provides information with minimal effort.

APPENDIX: AN EXAMPLE OF THE APPLICATIONS OF IRIS

The above is perhaps a bit vague: Consider the example that the student has acquired some data. He has measured the pressure above a liquid at different temperatures. The theory predicts that the log(pressure) vs 1/Temperature will be a straight line. The following steps illustrate the use of IRIS in testing this theory.

1) The student defines two vectors as PRESSURE and TEMPERATURE (names are arbitrary).

2) The actual data recorded is entered into the variables using the spreadsheet facility.

3) A new vector representing the log of the pressure is defined as LOG(PRESSURE). (IRIS performs the log calculations and places calculated values in the new vector)

4) The temperature is also transformed by defining a new variable called 1/T and defining it as the inverse of TEMPERATURE.

5) The student then defines a new vector called FIT which is defined as the least squares fit using 1/T and LOG(PRESSURE). IRIS will bring up a least squares definition screen to allow definition of additional variables such as the slope and intercept of the fitted vector and error terms.

6) The student can call up the scalar value table to obtain the slope and intercept of the fitted line

7) To graph, the student merely fills out the graph definition screen to plot both LOG(PRESSURE) and FIT as functions of 1/T. Once the graph is plotted on the screen, it can be dumped to a printer or a cursor can be activated to read data points from the screen.

8) At the end of this process, the student not only has tested the theoretical model, but has understood the model by defining it in IRIS. Tangible results include: graphical printout of the data, valuables for the variables which characterize the relation between TEMPERATURE and PRESSURE, and the errors in these quantities.

MOLECULAR STRUCTURE GRAPHICS FOR UNDERGRADUATE CHEMISTRY EDUCATION

Presenter: **G. Farber** *Massachusetts Institute of Technology*

Coauthors: **G. Farber** *Massachusetts Institute of Technology*
B. Kohler *Massachusetts Institute of Technology*
H. A. Wilder. *Massachusetts Institute of Technology*
J. I. Steinfel. *Massachusetts Institute of Technology*

INTRODUCTION

A central part of teaching organic chemistry is illustration of three–dimensional concepts such as molecular symmetry and spatial rela– tionships between the parts of a complex molecule. This is traditionally done by means of two–dimensional blackboard and paper projections, aug– mented by "ball–and–stick" model kits. Both of these media have severe limitations. As a part of the Athena project on Computer Applications in Chemistry Education at M.I.T., we have been developing an interactive molecular structure graphics system based on PC ATs. This system will allow students to construct arbitrarily complex molecular structures using keyboard and mouse inputs, view space–filling models of their structures, and obtain quantitative information about molecular energies and geometries from a linked database.

The essential educational purpose of molecular structure graphics is to allow the chemistry student to display and manipulate a variety of structural representations of organic and other compounds with the same ease that one manipulates a physical object. In this fashion, a molecule ceases to be an abstract collection of letters and lines in a textbook and becomes a physical entity with characteristic size, shape, handedness, etc. Although it is possible for students to construct models of atomic structures from kits that are commercially available, such kits never give the student a feeling for the ways in which the shape of a molecule determines its chemical properties. Also, it is impossible for model kits to switch easily from one form of representation to another, and it is very difficult to compare models of complicated structures without the use of an enormous number of model components. The software described here is intended to allow the student access to an immense stock room of molecular components, thereby enlarging his or her experience while teaching the student to think about molecules in a new and more realistic way.

OVERALL COMPUTING STRATEGY AND IMPLEMENTATIONS

The system is based on a set of interactive graphics routines for construction, manipulation, and display of molecular structures [1]. The principal routines are:

(1) Construct a two-dimensional representation of a molecule, using a "kit" of atoms, bonds, and chemical groups. Items are selected, placed on the display screen, and connected with each other by a pointing device such as a mouse. Various additional features may be invoked, such as designating formal charge or chirality of an atom, or aromaticity of a ring. The completed structure may be stored in a .MOL file, or a .FRG file for molecular fragments.

(2) The two-dimensional structure created in the preceding step may be converted to a geometrically correct three-dimensional structure by invoking a Conversion routine which employs a table of interatomic distances and bond angles. In subsequent implementations, this first-order structure may be processed by a computational procedure which takes account of non-bonded interactions to yield accurate molecular conformations.

(3) The three-dimensional structure may then be displayed on the screen, in either vector ("ball-and-stick") or spacefilling format; hard copy of either representation may be obtained if desired. The displayed structure can be manipulated in space, using a mouse to control its position, angular orientation, or scale. This provides the user with insight into the geometrical properties of the molecule.

While the CHEMNOTE routines are designed for use by a professionalchemist working at a terminal or PC, in the educational setting they would typically be invoked by a prewritten lesson script which guides the user (typically a student in an introductory organic chemistry subject) through a series of questions and requests various structures, displays, or verbal responses. The lesson scripts, in turn, are produced with an authoring language ("MKLesson") which allows the instructor to create text, molecular structures, and a variety of responses to students' answers. Structure- checking and text-comparison routines will provide immediate feedback to students using the lesson scripts, if that is desired; otherwise, students' responses can be filed for subsequent examination by the instructor, or hard-copy printouts can be obtained and turned in, for example with laboratory reports [2]. With this authoring language, individualized lesson scripts can be easily created; in the teaching of introductory organic chemistry, for example, this flexibility is vastly preferable to "canned" lesson scripts.

The programs are all menu-driven, so that neither students nor instructors are required to be familiar with programming languages or even text editors in order to be able to use the system. All software has been written in the C language and compiled under DOS 3.0 for use on the IBM PC/ATs. We expect to be able to transport these routines to Berkeley 4.2 or some other variant of Unix with a minimum of difficulty, when the next generation of networked workstations becomes available from Project Athena.

RESULTS AND FUTURE PLANS

Although this project is still in a fairly early stage of development, students' reactions to this approach have been encouraging. The software appears to be easy to use, and the ability to create and examine fairly complicated molecular structures enhances both lecture and laboratory material in this subject. Future developments include implementation of a conformational analysis program for producing optimized three-dimensional structures [3], and eventual linking to chemical data bases providing access to spectroscopic, chemical-reactivity, and toxicological information.

ACKNOWLEDGEMENTS

The developments described in this report were carried out as a part of the Athena Project at M.I.T., with support from the Kenneth J. and Pauline S. Germeshausen Fund and the Camille and Henry Dreyfus Foundation. The graphics utilities were furnished by Polygen Corporation, who retain all rights in the "Chemnote" software. Numerous individuals have contributed helpful comments and suggestions, but we wish to thank Prof. G. Petsko and Dr. D. Ringe in particular for their enthusiasm and insight.

FOOTNOTES AND REFERENCES

*Present address: Bell Telephone laboratories, Murray Hill, N.J.

[1] "CHEMNOTE" software proprietary to Polygen Corporation, Waltham, Mass.

[2] A laboratory data acquisition and analysis system ("Athena Chemlab System") has also been developed as a part of this project, and is in use in the undergraduate laboratory subject.

[3] "MMIIPC: An Interactive Version of MM2 for Microcomputers", Quantum Chemistry Program Exchange No. QCMP-010.

NEW ELEMENTARY PHYSICS LABORATORY EXPERIMENTS

Presenter: **D. J. Kane** *University of Illinois at Urbana–Champaign*

Coauthors: **D. J. Kane** *University of Illinois at Urbana–Champaign*

U. E. Kruse *University of Illinois at Urbana–Champaign*

J. H. Smith *University of Illinois at Urbana–Champaign*

Four new laboratory experiments using IBM computers have been introduced in Physics 106, Mechanics, at the University of Illinois. Physics 106 is the first semester in a three semester physics course for scientists and engineers.

The IBM computers are used for data acquisition, processing,plotting and printing as well as student–computer dialogue. The new data acquisition techniques have made experiments possible which could not previously be done. For example, the students now explore the motion of a system with a magnetic braking force, and they examine collisions of two gliders on a nearly frictionless air track in much greater detail and with much higher accuracy than heretofore possible. The much more rapid data acquisition allows a nearly continuous real time display of the physical quantities like position, velocity, acceleration, momentum, and energy.

The data are acquired with a homemade data acquisition board which accepts the signals from photo detectors recording the motion of mylar tapes or disk encoders. The tapes or encoders have alternating black and clear stripes that produce a series of electrical pulses as they pass through the photodetectors. The data acquisition board counts these pulses to produce an accurate position measurement. Circuitry on the board also allows the determination of the direction of motion. The position of the gliders on the air tracks is determined to an accuracy of 125 microns (1/8 millimeter) or the angle of a disk to an accuracy of 3 milliradian.

During the experiments, the computer asks the students to make important predictions before the data runs. Only after they have answered the questions correctly can they take the actual data. The computer program tries to be helpful in handling student responses: if some simple typing errors occur (such as an extra decimal point) the computer points to the place of confusion and asks for correction. If the answer is off by a factor of ten or hundred, the computer furnishes a helpful hint, During data acquisition, the positions are plotted in real time so the students can see the correlation between the actual motion and the data plots on the computer screen. They then select the portion of the data they wish to process. Only after each student has processed the first few data points correctly (such as calculating the momentum or energy) will the computer handle the remaining data points automatically.

There are five laboratory periods scheduled with the computers. The first gives the students a chance to practice using the computer. This introduction is followed by four experiments each in a two hour laboratory session. The first three experiments involve the motion of gliders on nearly frictionless air tracks. Students explore Newton's Second Law (F=m*a), Energy Conservation, and Collisions. The fourth experiment uses a torsional pendulum for the study of an oscillating system.

In the first experiment the students examine the motion of two systems to study the application of Newton's Second Law. The first system is a glider which is accelerated by a tape connected over an air pully to a small hanging weight. After predicting the expected acceleration the student runs the experiment and observes the position of the system at selected time intervals (usually chosen to be every tenth of a second). From the positions he is asked to find the velocity and the acceleration for the first few intervals. The computer then fills out the remaining values allowing the student to examine the variations with time and compare them to the expected results. The second system adds a magnetic braking force. The student first determines the characteristics of this "unknown" force and then confirms that by adding this new force to the force from the hanger he can get excellent agreement for the acceleration of the system by applying Newton's Second Law.

In the second experiment the students study the conservation of energy for three systems. The first system is identical to the simple system already examined in the first experiment. In the second part the weight hanger is replaced by a spring and in the third part the track is inclined slightly so that the students can observe the interchange of potential energy in the spring with the kinetic energy of the glider and the gravitational potential energy of the glider. With the use of the computer a careful student can confirm the conversation of energy to a few percent.

The third experiment studies elastic and inelastic collisions between two gliders moving on an air track. For the elastic collision a spring is attached to one of the gliders as shown in Figure 1. This arrangement,

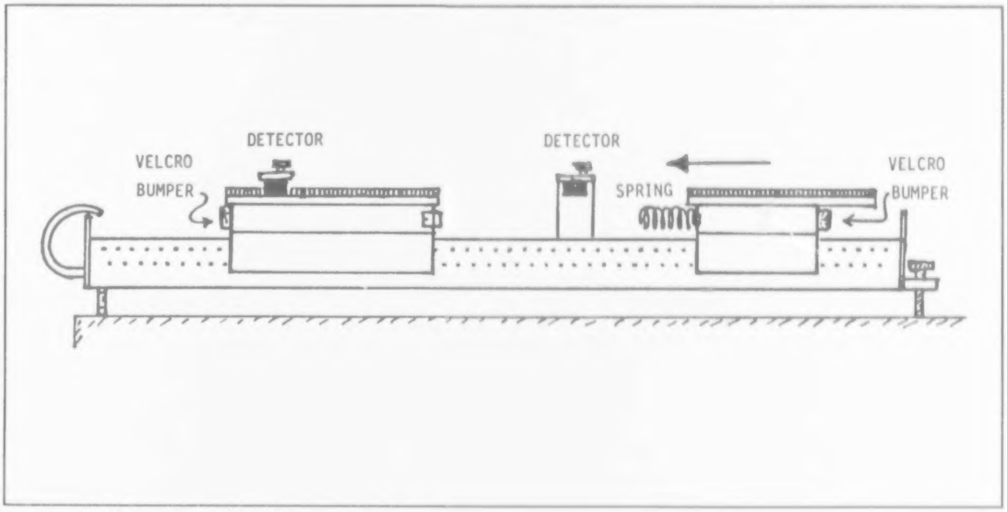


Figure 1.
Apparatus for Collision Experiment

together with sampling the position every .05 second, permits the student to observe the interchange of momentum and energy during the collision. Figure 2 shows an actual screen image taken from the experiment and shows how the various forms of energy change over a one second interval which encompasses the collision. Dots, dashes, and a horizontal line have been added in the figure to guide the eye. The students can see that during the collision the total kinetic energy of the two gliders decreases, however the potential energy stored in the spring between the gliders makes up for this loss so that the total energy is nearly

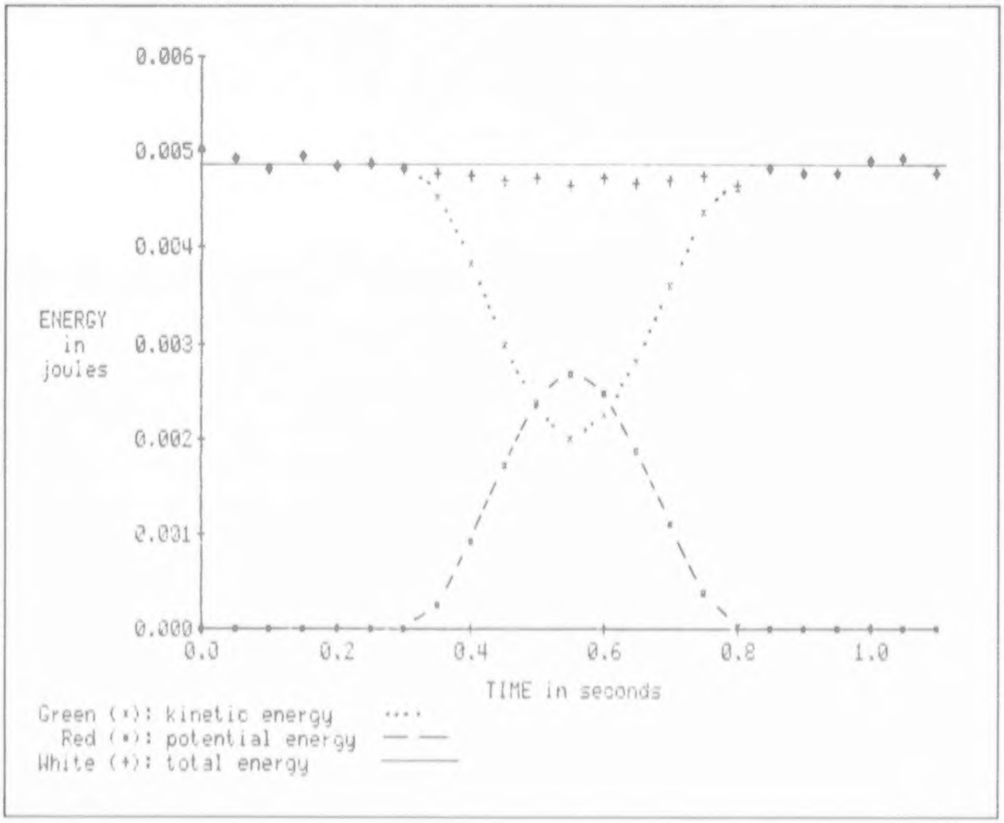


Figure 2.
Kinetic, potential, and total energy during elastic collision.

constant throughout the collision. It is also apparent from the figure, that in this case the spring constant was not correctly determined. A slightly larger spring constant would increase the values of the spring potential and total energies during the collision, yielding an improved fit to the horizontal line expected from conservation of energy.

In the fourth experiment the students use a torsional pendulum for the study of an oscillating system. They first observe the motion of the oscillating disk. By making measurements of the angular displacement plotted on the computer screen they are challenged to determine the parameters in the theoretical formula for the angular displacement. The computer then plots the curve derived from their parameters and compares directly with their experimental points. When they have a satisfactory set of parameters the comparison is extended to include the angular velocity and the angular acceleration. A plot of this comparison is shown in Figure 3 from which the students can easily see the relations among the three quantities. The experiment continues with the determination of the physical parameters of the system: the moment of inertia of the disk and the stiffness or torsonial constant of the wire. These values permit the student to make detailed comparison between the observed motion and the predictions from Newton's equations for rotary motion.

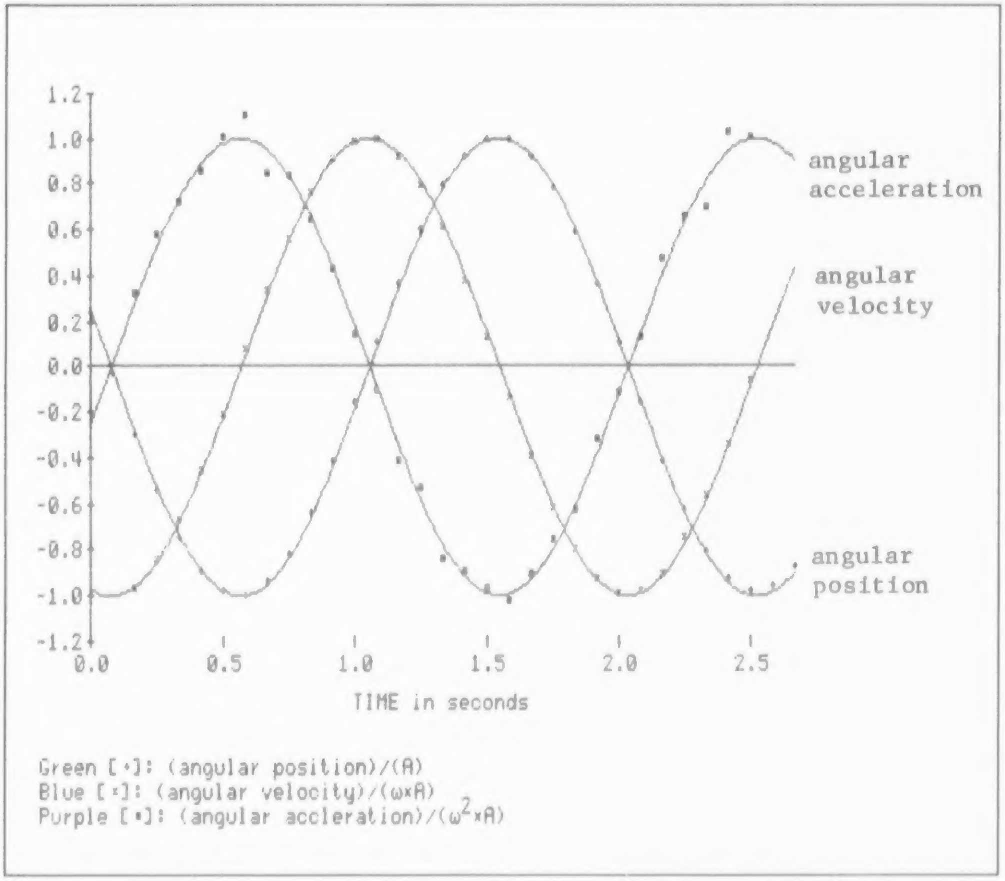


Figure 3.
Angular position, velocity, and acceleration for torsional pendulum

The TenCORE computer language has been used to write the main routines in the program with data acquisition, rapid plotting, and printing of tables and screen displays handled in assembly language. The students therefore have a convenient record of their measurements and theoretical comparisons which they use in their writeups. In the writings they refer to these records to answer questions reviewing the physics that was tested in the experiment. This system has turned out to be very efficient and flexible. Several changes have been made in response to suggestions from students and staff during the first use in the fall semester 1985. Further improvements are being made during the spring semester 1986. At this time the system is being used by about 1200 students.

THE USE OF EXPERT SYSTEMS TO TEACH VETERINARY DIAGNOSTIC DECISION-MAKING

Presenter: Ronald D. Smith *University of Illinois*

The College of Veterinary Medicine, University of Illinois, offers a 4-year curriculum leading to the Doctor of Veterinary Medicine, or DVM, degree. Although a variety of career opportunities are open to veterinarians, most students come because they want to learn to become veterinary practitioners. The size of the veterinary medical knowledge and procedure database has spawned teaching methods which are sometimes inconsistent with the demands of the profession. First, it has fostered fragmentation of subject matter, deemphasizing the holistic approach to medicine. Second, the knowledge-intensive curriculum favors rote memorization of medical facts and cause-effect relationships over development of diagnostic problem solving skills. Third, medical knowledge often is not organized in a way that facilitates problem solving. It is relatively easy to find the signs and symptoms of a given disease, but difficult to find the cause of a unique set of signs and symptoms.

The first 3 years of our curriculum consist largely of lectures and laboratories designed to teach students about the biology of a variety of animal species, the diseases which can befall them, and the techniques that can be used to diagnose and treat them. This process is one of "information transfer". During the fourth year students are expected to apply the information that we have given them to actual problem-solving in the Veterinary Medicine Teaching Hospital. Here students come to face with the public for the first time. This is where students must "turn around" all of the cause ----> effect information that we have given them, and learn to use patient findings to determine the identity of a particular disease. This can be a traumatic and confusing experience, particularly for those who are still in the "memorization mode."

The objective of this project is to help veterinary professional students develop the problem-solving skills needed in the practice of veterinary medicine. Problem-solving exercises should be introduced into the curriculum as early as possible so that, by the time our students enter the fourth year of the curriculum, they have already developed an understanding of and appreciation for good diagnostic procedures.

We are using 2 approaches to the problem. Both take advantage of the computer's ability to organize large amounts of data into meaningful information. The first, which may be referred to as "management aids", teach students how to identify medical problems that require attention from patient or herd medical history and physical examination. "Diagnostic aids" continue where the management aids finish, i.e., they start with a particular medical problem, guide the student through proper history taking, physical and laboratory examination, and suggest possible causes for the condition. These aids help students to learn standard diagnostic protocols, and teach them to be thorough.

One way in which diagnostic aids can contribute to our curriculum is evident if we review the duties of our students in the veterinary medicine teaching hospital (VMTH). The process begins with the student retrieving the patient folder which contains the basic patient information and all of the medical record forms that will be used. The "Admission Record" and "Case Summary Sheet" list all of the possible patient findings that the student is expected to look for. The patient is then taken into the examination room where patient history and physical examination findings are recorded. At this point the student must utilize the patient history and physical exam findings to formulate a "Problem Cue List" and "Initial Plan". The "Problem Cue List" includes general structural or physiological abnormalities, with the exception of specific disease entities. The "Initial Plan" includes the rule–outs (r/o) to be considered, i.e., a list of specific diseases which the patient may be suffering from, and additional workup (w/o), treatment (Rx), or client education (client ed).

In many veterinary schools, including our own, students often use acronyms to help them formulate a rule–out list of possible diseases in terms of their pathophysiologic mechanisms (Table 1). This mental exercise requires that the student recall all of the information acquired earlier in the curriculum, and sort through it to determine possible disease entities.

Table 1: "D–A–M–N–I–T" Acronym of Pathophysiology

D — Degenerative disorders
A — Anomalies
— Autoimmunity
M — Metabolic disorders
N — Neoplasia
— Nutritional disorders

I — Inflammation (infectious or noninfectious)
— Immune disorders
— Iatrogenic disorders
— Idiopathic disorders
T — Toxicity (endogenous or exogenous)
— Trauma (external or internal)

This process of information recall is completely opposite from what was expected of the student during the previous 3 years of the curriculum. Rather than recruiting the signs and symptoms of a given disease, the student must now recall all of the diseases which can cause a particular set of signs and symptoms. The student must also know which clinical findings are of greatest diagnostic value. This is the point at which a computer–based diagnostic decision–making program can have a significant impact on the development of good diagnostic decision–making skills.

The case record data is being used to guide us in the selection of diseases to be included in decision–making programs. Information from every clinical case is recorded on an abstract form and then transferred to a mainframe computer on campus. The site and identity of each diagnosis on a patient (there may be more than one) is recorded onto the abstract form. Information in this database is downloaded to an IBM PC/AT where it is sorted to learn what medical conditions students are most likely to encounter during their clinical rounds. For example, almost 50% of diagnosis in our VMTH deal with organs of special sense (eyes, ears), skin and digestive tract. Parasites and microbial infections account for more than 50% of skin problems. This information can be used to assure that none of the common diagnostic problems are omitted from our decision–making programs.

Several different approaches to medical decision–making are available. These include rule–based and probabilistic systems, decision trees (medical algorithms), and knowledge–based systems. Our college has chosen to use a variation of the knowledge–based system from "PKC Corporation", which is essentially a relational database, for application in clinical settings. We have found that this particular package is compatible with the problem–solving approach embodied in our own "Problem–Oriented Medical Records". Each diagnostic program will accommodate up to 600 history, physical examination or laboratory findings, and can support up to 300 possible management options or diagnosis. The program will run on any IBM PC (portable, PC, XT, AT) with 2 floppy disk drives or a hard disk drive at least 256 Kb of internal memory. Characteristics of the PKC diagnostic software are listed in Table 2.

Table 2: Attributes of the Knowledge-Based System

A. More findings ---> more diseases.
B. "Cause" output format can be personalized.
C. Ranking of probable causes.
D. Suggests additional examination procedures.
E. Comments in predictive value of findings.
F. Suggests follow-up diagnostic tests.
G. Comments on prognosis.
H. Indicates treatment.
I. Lists bibliographic references.

Programs currently under development are in the general area of diagnostic "Rule Outs", and specific areas such as neurological diseases, surgical options, avian diseases, and porcine and bovine infertility.

Besides helping students with problem-solving in the outpatient clinic, an additional objective of this project is to teach problem-solving skills earlier in the curriculum. Although the students do not have enough training to deal with the entire spectrum of diseases of domestic animal species, exercises have been developed which focus on particular problems. An example is the use of a problem-solving program in a course entitled "Nutritional Aspects of Large Animal Diseases", which is offered during the third year of the curriculum. This program was originally written for use on our PLATO teaching computers. It has since been translated for use on the IBM PC's. Students are first introduced to the program using a video projection unit in the typical classroom setting. Next they are given problem sets consisting of actual production and feed data form problem farms. They must utilize this data to determine what sort of problems exist, and whether feed imbalances are contributing factors. Students also use the computers in their midterm and final examinations. Since the program has been adapted to run on the PCs it can be used on PCs in other veterinary schools. The portability of the program has also enabled extensionists to carry it to the continuing education programs."

In summary, the IBM-EXCEL equipment grant to our college of veterinary medicine has enabled us to present traditional course material in a way that enhances the development, and the impact of the project in the college will continue to expand. It is important to emphasize that we are not adding new courses to the curriculum, nor are we making the students learn to do something new. We are simply using the computer to help our students learn the science rather than the art of medical care.

USING CAMPS TO TEACH MEDICAL PROBLEM SOLVING

Presenter: William Schwartz *University of Pennsylvania*

copyright © DACLS Software, 1986

CAMPS, Computer Assisted Medical Problem Solving, was designed to provide faculty members with an authoring system to develop simulated medical cases to teach and evaluate medical problem skills. The objectives for the CAMPS design were:

1. A system for faculty members who did not have programming skills which would allow them to author medical case simulations.

2. The product, a CAMPS case, should closely reflect a clinical encounter, allowing the user an opportunity to ask history questions, obtain physical findings, order laboratory tests and consultations and begin treatment.

3. The system should have editing features and utilities which helped the author prepare a case in an efficient manner. The goal for authoring time was one hour.

4. The simulation should contain easily understood menus which guide the author through the case creation process minimizing the need for manuals.

5. The system should include a scoring system which would automatically grade the student record and include information which allowed analysis of the student performances.

6. The system should be include ability for editing, additions and deletions of case items.

7. The system should be run on standard IBM equipment without the need for extra peripherals.

8. The product should not create negative feelings from the author or case user because of delays in responding to choices, unavailability of desired evaluation and treatment options, and scoring system.

9. Provide a method for research and evaluation of the problem solving skills.

The design plan for CAMPS consisted of development of a system which could generate medical cases that demonstrated examples of clinical conditions which were part of the course objectives or important problems which were seen infrequently. A number of cases would be written so that trainees could see either a range of problems or variations of the same chief complaint. These exercises would increase the students' experience so that they could be more familiar with the clinical problems when they worked with real patients, and ability to obtain information to detect differences in clinical problems. The cases could either be used as a tutorial which included a discussion at the end of the exercise or as a evaluation method where the trainee would be graded on the completeness and correctness of the diagnostic process. The chosen items could be saved for further analysis to detect problems in data gathering, assessment or management. The clinical problems would be placed in a centralized location such as a library, so that the users could evaluate a patient at their convenience.

EXPERIENCE WITH THE CAMPS SYSTEM

Although other systems which presented clinical material for student's use existed, they required many hours for development and financial support for program maintenance and storage. Clinical faculty, who saw the need for computer based instruction, did not have the time or budget to support this activity. The combination of careful planning for the busy faculty member plus utilization of a microcomputer allowed this project to reach the goal of a low cost, efficient teaching method.

CAMPS originated from the card system of Rimoldi who developed a method of analyzing the problem solving pathways using a pack of cards from which the student picked options of history, physical or laboratory items. The pathway of problem solving could then be analyzed by looking at the content and selection order of the cards. This system was further refined by Helfer et al who used a computer to score the choices. The microcomputer provided the technology to expand the basic concept into a larger datafile system with enough memory to hold text and to recall information for analysis.

CAMPS (version 1) originally designed in 1978, employed available technology and memory(64K). This primitive system design underutilized the power of the computer. The evolution of the program for the IBM format reflected a more sophisticated design and incorporated programming skills to now include important editing capabilities, file storage and use of the function and arrow keys. Problems with logic and case flow were corrected based on comments by the users during the development phase.

The present version 3.2 is used in over 30 medical and nursing schools. At the University of Pennsylvania, CAMPS is used extensively in the Department of Pediatrics introductory course for teaching and evaluation. A library of cases is available for independent study and for assignments for seminars on problem solving techniques. In these meetings, the students come to class after evaluating an assigned group of cases. They use these data for discussion about facets of problem solving, development of the differential diagnosis, history taking strategies, rationale of laboratory test selection and final diagnosis. These discussion are either Socratic or small group debates where the students are assigned to groups of four where they decide on their approach to the problem and then prepare their justification for their choices. This method allows more students to participate in the dialogue and gives the instructor more information about the class's problem solving technique. In one school, the students were given a choice of either writing a paper on a disease or making a CAMPS case. The majority chose the CAMPS system which required them to develop a history and physical examination as well as describe the abnormal laboratory tests and all possible treatments which would affect the disease.

Evaluation of student problem solving skills using CAMPS has shown it to be an effective testing method. After students interact with the case, the record of their choices are examined by the instructor who can identify certain profiles of problem solving skills. Since the correct diagnosis is only part of the scoring formula, (the others are cost, treatment, laboratory tests, history and physical items and consultations), students may know the right answer but still have to show how they solved the problem. This profile offers more information about the thought processes than could be retrieved from a multiple choice test.

Cases made with CAMPS have been used to monitor the benefit of post graduate courses. Attendees are asked to work through the cases after attending the lectures. Faculty can then review the choices of evaluation and treatment to see if the new knowledge is being incorporated into the practice patterns of the audience.

EVALUATION OF CAMPS

Program evaluation indicated that the students accepted the cases as an important part of the teaching program. Attitudinal surveys from over 200 students at Penn indicated that over 85% of the students considered CAMPS a positive experience. They felt that it was fun to use, improved their sensitivity to test choice and to the costs of a medical evaluation and was a valuable learning method. Experience from users in over 30 medical, nursing and allied health schools has been favorable with most variation reflecting the interest in the faculty and the access to use in the curriculum. In one medical school, students who interacted with CAMPS scored higher on the final examination than control groups who had the same course and clinical assignments but did not use CAMPS. This indicated that the more extensive clinical experience in the computer group improved their general evaluation. A study on the scoring method indicated that the cases needed to be difficult to be valid, i.e. that when the diagnosis was difficult or obscure, the faculty scored higher than the students. There was no significant difference when the cases were "easy." CAMPS is now used in our department to define problem solving skills, to practice evaluations and for final examination in clinical courses.

Faculty have had mixed reactions to developing cases. Many have authored cases and instituted CAMPS as part of their programs. The recent administrative interest in computer based instruction should expand the usage. Students are less fearful of computerized learning since they have had exposure in past educational experiences. The most success has been where the author has access to a course where the cases can be used rather than write cases and hope that some course will use them. Others, despite the efficiency of the authoring system, found it difficult to transform their ideas to an actual case. Some faculty resist the concept of computers as teachers or evaluators. This year at the University of Pennsylvania, a microcomputer laboratory is under construction which will create an environment for computer based instruction. Additionally, a course funded by the IBM Educational Development Grant will be conducted to instruct faculty in creating cases for clinical courses.

SYSTEM DESIGN

Major priorities of the objectives were minimal demands on faculty time and creation of a product that was flexible and easy to use. The busy clinical faculty member with limited time and computer skills was the principal audience of the system. The authoring system has 6 modules:

1. Editing
2. File management
3. Student monitoring

4. Creation of the student diskette
5. Printing of files
6. Printer configuration

Each module has utilities, selected from a menu, which perform important functions in case creation and scoring. Three main files are used to develop a case. Fig 1. The first file, the perfect patient is a listing of over 500 items which describe a typical history, physical, available laboratory tests and treatments of a normal person. Files can be made for a patient of any age or sex. The case file or the "sick patient," results from the substitution of about 30–50 items in the perfect patient file. For example, if a case of diabetes is being developed, items such as blood sugar and diet are changed whereas examination of the ears or chest X–ray do not have to be changed. The case file is then merged with CAMPS routines to create a student disk, the final CAMPS product.

Fig. 1

```
    30–50 items > > >   CASE FILE  > > > >   STUDENT DISK
        ^                                         ^
        ^                                         ^
        ^                               CAMPS ROUTINES
 PERFECT PATIENT                                  ^
        < < < < < <  AUTHORING SYSTEM > > > > > >^
```

Since the authors work consists of substituting text in the perfect patient file to make a case, the authoring systems have extensive editing functions which make the program more efficient. Insertion, deletion, search routines help the author make appropriate changes.

The CAMPS authoring system leads the developer through editing options and finally the case creation. The system is menu driven which follows the logic of case development, review of the perfect patient, substitution of case specific items to make a patient file and finally the case creation module which merges the patient file with CAMPS routine into the student diskette, the CAMPS product. Although a manual is provided, the system can be used by following the directions on the menus.

The design of the system was based on the premise that the user should not get angry with delays, rigidity, or lack of direction such as how to exit the system or return to higher level menus. Prompt responses after choices are entered help maintain the flow and rhythm of the case. Use of menus rather than free language, although potentially restrictive, maintain the pace of the encounter by avoiding long searches for free language entries. The editing system allows for changes even after a case is developed so that, any items felt to be important by the user can be added to existing cases. Menu cursor activity and uncluttered screens keep the user's eye in the correct area for indicating the next type of selection.

SCORING DESIGN

The scoring system gives a composite numerical evaluation plus a breakdown of the various subscores which include: number of history and physical items, weighted scores of laboratory tests, number of important and unnecessary treatments and total cost and correct diagnosis. The scoring system considers seven subscores of the problem solving process, history and physical items, laboratory scores, correct diagnosis, cost, treatment, and consultations. The author assigns a score from −10 to +10 for each laboratory item reflecting the importance, danger and cost of the item. The author also establishes the weight of each subscore reflecting the contribution of each group to the final grade. Since there is a tendency for committee decisions to reflect the bias of the most outspoken member or what the committee thinks is theoretically the right way, this strategy neutralizes these biases by requires the experts to work through the case without discussion or prior knowledge of the diagnosis. CAMPS saves each of the expert's choices and develops a panel score. For example, if the author did not feel a chest X-ray was necessary but three of five experts ordered an X-ray, then the student who orders this test receives credit while the one who did not loses points. The same is true for treatments where the student is compared with the majority selections.

The score is calculated from a formula:

SCORE= $a(H) + b(P) + c(L) + d(T) + e(C) + f(\$) + g(D)$ where a,b,c,d,e,f,g are the weights for each subunit and H = History items; P=Physical items; L=Laboratory T=Treatments; C= Consultations; $= Cost of evaluation; D=Correct diagnosis. Each subunit is a ratio of the student score over the expert panel score. The final grade is a ratio reflecting

how close the students evaluation
approximated the expert panel.

The student is then judged by comparing the performance with the expert consensus. More important than the correct diagnosis are the other factors which enter into the problem solving process.

MONITORING

The student and faculty can retrieve the summary of the score or a complete listing of all the choices. From this listing, a problem solving profile can be generated using forms prepared for this purpose.The scoring system detects the student whose general evaluation pattern resembles the faculty experts and those who either streamline the workup so that the evaluation is incomplete or those who are overly compulsive which translates into inefficiency and expense. Students who think their diagnosis was correct have the ability to enter a protest. This monitoring module permits the faculty to see the entered diagnosis and decide if the entry is acceptable. If so, the score is recalculated to reflect this change. Monitoring segment also includes the options to print the listing of student scores, essentially grading the class immediately after the last student takes the test. The author can use monitoring to change the number of experts that are required to establish the reference score, to change the weight of each subsection and to put the case in the lesson mode, where the correct diagnosis and discussion is displayed or in a test mode where the diagnosis is entered by the student who in turn is given only the cost of the evaluation.

STUDENT DISK

The student disk, created by the authoring system, centers on a Main Menu which lists the major options of the diagnostic process, History, physical, laboratory, treatment, consultation, and disposition. This allows the student flexibility to move through the evaluation process to make a diagnosis and begin treatment. All choices are made by moving an arrow to the appropriate level on the various menus and pressing the enter key. Only the diagnosis is enter in free text. In contrast to paper systems, CAMPS gives the student unlimited choice of pathways to take in solving the problem.

TECHNOLOGICAL DESIGN STRATEGIES

UCSD PASCAL was chosen since the modular structure promoted efficiency in programming and rapid response time for the user. Data are held in memory as a tree structure composed of a nested list which gives it flexibility and allows rapid access to a large body of information. Items are recalled by use of pointers. For example, to find the patient's medication, one must choose "History" which presents major categories of the history from which "Past Medical History" is chosen and leads to "Medication" where the text is displayed about the patient's present therapy.

Fig. 2

```
HISTORY---PRESENT ILLNESS
          REVIEW OF SYSTEMS
          PAST MEDICAL HX----ALLERGIES
                             -TRAUMA
                            -MEDICATION----"TAKESASPIRIN
                                             FOR HEADACHE"
                                            --------------
                                              displayed text
```

The authoring system establishes the labels for the nodes of the branch as well as the text of the last segment. This tree structure allows for expansion and contraction of the patient listings in the editing process. If the case requires modification of the perfect patient listings, these items are easily added.

EDUCATIONAL FEATURES

The utilities built into CAMPS assisted in keeping the work for the faculty member at a minimum. The perfect patient design enables the author to load into the system 500+ items and make from 30-50 substitutions to create a case. Case editors facilitate the entry of the new text.

Adhering to the one hour goal for authoring kept the accessory features to a minimum. Although visuals have an attraction to the viewer, creation of these displays add considerably to the authoring task without proof of educational value. Likewise, adding comments about each choice and invoking rules denying certain choices if selection criteria were not met present a higher level tutorial system and can, in some cases, add hours and weeks to the case creation process. By respecting the initial objectives, these attractive options were not included in the authoring system in which to keep the authoring time requirement to 1 hour.

Flexibility in choices closely simulates the actual clinical encounter. The student can select any of the major items which include:

HISTORY
PHYSICAL
LABORATORY
TREATMENT
CONSULTATION
DIET
DISPOSITION

These major groups may have a number of branches. For example, if faced with a patient in coma, the first action may be to take a blood pressure or order some laboratory test rather than start with a history. A major premise of this format and scoring is there is no one correct pathway to get the right diagnosis and treatment. The freedom of selection gives the student the opportunity to explore the case in an individualistic manner thus exposing his problem solving skills. CAMPS gives the instructor the chance to analyze these patterns.

The expanded memory of the IBM personal computer allows the saving of student performance file for review by the faculty. Each student enters his name which identifies the file where the choices are saved for review by the faculty. In addition to the score, information about the number of choices, cost and laboratory selections are available for faculty review. To see how the student performed, the instructor uses the authoring system– monitor option. The student disk is inserted and the performance records are immediately displayed– truly electronic grading. (see Figure 3.) To see an individual's record, the arrow is moved to the name and the summary is shown. (Fig 4.). Finally all the choices are displayed. All these records can be printed. Review of this selection file enables the faculty or student the opportunity to analyze and discuss problem solving skills.

SUMMARY

CAMPS, Computer Assisted Medical Problem Solving, is an authoring system which allows the creation of patient simulations. CAMPS enables a faculty member without programming skills the tools to develop a case within one hour. Features of the design include: perfect patient file, a listing of 500+ items which describe a normal person; use of tree structured memory which facilitates retrieval and editing of items; monitoring utility which allows establishment of scoring values, instant grading and printouts of performances and flexibility in selecting choices. The objectives of the design goals were achieved to enable the system to function in more than thirty medical schools. Student acceptance is very positive. Evaluations are now being conducted to determine the value of CAMPS in teaching problem solving.

Figure 3.

Monitoring function of the Authoring System. All student records are displayed with data about the performance. Individual records can be examined in more detail. See Figure 4.

```
Use to select:  Del to delete: .. PgUp, PgDn, Home, End to move;  Q to quit
           All entries              |
Name       Class     %      $   dx? |
           student  56    300no     |
           student  54    270no     |
           student  59    340yes    |
           student  34    670no     |
           student  71    253yes    |




F1=Show all        Print entry=F2|
F3=Show students   Recalc score=F4
F5=Show staff      Misc.=F6      |
F7=Show experts    Change disk=F8|
F9=Print list      Remove file=F10
```

Figure 4.

Detailed information about this student's performance. The large number −1 lab tests contribute to high cost of this evaluation in comparison with the others in this section. This student did not have a focus but skipped around looking for an abnormal test.

```
Use to select: Del to delete: .. PgUp, PgDn, Home, End to move: Q to quit
           All entries            |
Name       Class    %    $   dx?  |Name:  Port       Class:  student
           student 56   300no     |Score:  34%      Cost $670   Diag. ok?:  no
           student 54   270no     |Laboratory point value distribution:
           student 59   340yes    |-10 -9 -8 -7 -6 -5 -4 -3 -2 -1 0 1
           student 34   670no     |-----------------------------------
           student 71   253yes    |  0   0   0   0   0   0   0   0   0   0  0  0
                                  |  2   3   4   5   6   7   8   9  10
                                  |-----------------------------------
                                  |
                                  |Treatment value distribution:
                                  |  Necessary  Unnecessary  Harmful
                                  |      0           2          2
                                  |Hist:  53;  Phys:  33;  Consult:  0
                                  |This person claims the diagnosis is
F1=Show all      Print entry=F2   |SIMPLE FEBILE SEIZURE
F3=Show students  Recalc score=F4
F5=Show staff     Misc.=F6        |Show this person's choices?
F7=Show experts Change disk=F8
F9=Print list   Remove file=F10|
```

INDEX BY PRESENTER / AUTHOR, VOLUME III

Presenter / Author	Page
Anderson, Gary F.	114
Atkinson, Daniel E.	81
Barkley, David S.	81
Boh, Larry E.	77
Bolger, Michael B.	2
Bradley, Joan	85
Brauth, Steven E.	16
Clarke, Steven G.	81
Farber, G.	120
Gardy, Martin	93
Godley, Paul	85
Godley, Susan	85
Hagamen, W. D.	93
Kane, D. J.	124
Kohler, B.	120
Krull, James	85
Kruse, U. E.	124
Lamb, Sandra I.	81
Lomax, Anthony	109
Ludden, Thomas	85
Moore, John	85

Presenter/Author **Page**

Mulligan, Ed 72
Niklas, Karl J. 65
Pitterle, Michael E. 77
Rees, Douglas C. 81
Robinson, Bruce H. 114
Sagrave, Rosalie 85
Schultz, Larry W. 77
Schwartz, William 147
Smith, J. H. 124
Smith, Ronald 142
Steinfel, J. I. 120
Stixrude, L. 131
Taylor, Edwin F. 41
VanderVen, Ned S. 29
Walker, Russell C. 10
Wenk, H. R. 131
Wiederholt, Joseph B. 77
Wilder, H. A. 120

INDEX BY UNIVERSITY, VOLUME III

University	Page
Carnegie–Mellon University	10 29
Cornell University	65 93
Massachusetts Institute of Technology	41 120
University of California, Berkeley	109 131
University of California, Los Angeles	81
University of Illinois	142
University of Illinois at Urbana–Champaign	124
University of Maryland	16
University of Pennsylvania	147
University of Southern California	2
University of Texas at Austin	85
University of Washington	72 114
University of Wisconsin, Madison	77

IBM

Academic Information
Systems Proceedings

1986 University AEP Conference

Volume IV:

TOOLS FOR LEARNING

April 5 - 8, 1986
Sheraton Harbor Island Hotel
San Diego, California

CONTENTS, VOLUME IV

Title **Page**

A Generalized Videotex Laboratory 2

A Practical Authoring System/CBI with a Video Digitizer ....

An Administrator's View of AEP Program Operation and Evaluation 18

Beyond AEP 23

CMU Tutor:An Integrated Programming Environment for Workstations 29

Computer Augmentation of Flight Simulator Training 38

Computers in Residence: Social and Managerial Issues 54

Convivial Computing at the Forbes College Computing Center 60

Designing Local Area Networks for the Academic Environment 65

Innovations in Higher Education: A Multimedia Device ....

Networking to Support Education Objectives 76

PDS–A General Authoring System for Development of Simulations 83

Report on Brown/CMU/MIT Window Manager/Environment Convergence . 92

Scriptwriter II: From Vaporware to Freeware 98

The Challenge of Qualitative Evaluation 102

The IBM PC Network: An Introduction to Managing It Wisely ....

Tools for Learning at the Interactive Computer Graphics Lab. 152

Towards an Integrated Design System for the IBM AT 156

You Can Get There From Here 170

INDEX by Presenter / Author, Volume IV 202

INDEX by University, Volume IV 204

A GENERALIZED VIDEOTEX LABORATORY

Presenter: John E. Bowes *University of Washington*

GOALS

This project had as its goal the creation of an instructional lab which would demonstrate some of the possibilities of electronic publishing or videotex. Videotex, as many of you may know, is a method of compressing and encoding simple graphics and text for transmission over voice grade telephone lines. Since 1980, a de facto standard for videotex has existed in the U.S. and Canada, called North American Presentation Level Protocol or NAPLPS. Its capabilities to transmit limited color, graphics and text through standard telephones to specially adapted home televisions and personal computers represent the boundaries of a novel media form. For students and researchers of journalism, it is a medium which differs from the traditional in a number of key respects:

1. It is sequential, presenting information in related frames of text and graphics. Logical choices must be made by the user who is no longer a passive recipient of information but has an active role in seeking it out and structuring it through the interrogations made of the system. Students who wish to prepare materials for this system must change some traditional journalistic practices to accommodate limited capacity frames, the ability to select detail from menus and the prospect of using limited animation.

2. It is 2–way, a fundamental difference from the "one message for all" character of traditional media. Students must recognize specialized interests within their audience and accommodate these in the menu choices allowed.

3. It is rapidly changing. Information contained in the system can be continuously updated — market forecasts, weather, sports. It is a medium especially suited for the tabular display of statistical information. Students must thus create a structure of frame sequences and of explanatory material that can accommodate continuous updating of time sensitive content.

4. Modal aspects such as typeface, size, colors and sophistication of graphics are limited — some would say crude compared to print and standard broadcast media. Students must learn to use these capabilities to explore the limits of this medium.

5. Videotex systems have had a difficult time reaching profitability or mere break–even. The search for successful formulae of content, customers and cost is still young. We wanted a system which could record and display use statistics of some detail. Students need to learn the readership needs and marketing options for this medium.

STRATEGY

In short, we wanted an environment conducive both to student training in journalistic skills for this medium and to research endeavors. At the same time, we wanted a realistic environment which offered us the capability of a full, working videotex system. We wished it to be a basic tool for many purposes ranging from a faculty research tool to the basis for day–in, day–out CAI and journalism instruction. Thus:

1. A system controlled logically by a simple authoring language, similar to PILOT.

2. A system responsive to user inputs -- mimicking a 2–way capability characteristic of videotex.

3. A system which would accommodate multiple users simultaneously.

4. A system with all standard NAPLPS attributes.

5. A system able to record the artifacts of its use for each user and permit efficient aggregation of data obtained.

6. A system with basic software utilities, such as a debugger, which would simplify the programming environment sufficiently for non–programmers to use.

IMPLEMENTATION

Our strategy was based on reconciling our goals with resources and skills. Early in 1985, the Olympus Program and an equipment gift program of AT&T were utilized to provide the equipment and some programming support. From AT&T we applied for and received a mini–based NAPLPS frame creation system. From IBM we received an XT, some programming support and access to an instructional LAN of graphically able PCs. Our task was to interconnect the XT with the AT&T device which was capable only of creating frames, not of controlling their sequence through programmed instructions. The XT was to accomplish this through purpose-written software. A classroom LAN of PCs was to provide the multiple station environment we wished. Doing experiments one subject at a time or instruction one student at a time is not efficient.

We thus had a series of problems to solve.

1. Connection of the AT&T device to the PC-XT. The frame creation device has a special protocol for frame transfer to other machines. In this instance the "other machine" was anticipated to be an $80,000 AT&T mini computer. We were able to obtain a program FCTCOMXY which allowed the XT to mimic the necessary AT&T protocol for frame transfer.

2. Conversion of the PC-XT to a NAPLPS decoding device. Initially, this seemed to be a major problem. Several small Canadian firms provided Telidon software for the IBM-PC. But since NAPLPS is a more powerful implementation of the Canadian Telidon standard, there were certain problems in adapting the AT&T frame creation device to work in a Telidon environment. The solution seemed quite expensive until IBM itself announced the availability of PC- Videotex, software with most of the capabilities we needed. Ironically, since NAPLPS requires 8 colors simultaneously, the standard IBM color graphics card won't work properly. Rather, one with 3 bit planes, not 2, had to be provided. We use the Plantronics color board which gives us this capability. Our hope is to eventually have a proper driver for the IBM EGA which will allow that product to work as well.

The combination of the PC-Videotex software and the Plantronics board worked well. Minor problems were discovered in the handling of type fonts, since NAPLPS standards for typefaces cover only monospaced characters. The IBM holds to this criterion, preventing our use of more attractive fonts available in the AT&T frame creation device.

3. Providing Program Control of Frames. While the PC-Videotex software allows sequential control over frames, it has no direct provision for random access. Nor does it provide any programming language to allow more sophisticated selection such as menu options or conditional selection of frames (e.g., if error, go to....). PC-Videotex does provide "hooks" for external programs to select and display frames within a NAPLPS environment. A shell program written in turbo Pascal was created to utilize these hooks and provide the control options we sought.

RUNVTX provides a programming environment for frame control. Commands allows menu selection, repeating loops, subroutines, keyboard entry of information by users, and checking of inputs for errors.

It also allows the collection of data relating to individual use. One can associate with each frame selected by a user:

- Total times per session the frame was used
- Duration for first use of frame
- Cumulative time frame was used in session
- Sequence of first use (that is was it the first, third, etc. frame accessed)
- Number of use errors (hits wrong key, "wrong" being previously defined
- First Query –– that is if a single keystroke is requested by a frame, the first such selection is recorded
- Text of first read –– when user input is requested, that input (up to 40 characters) is recorded for the first such instance in a session

Given the number of frames possible in a given user sequence and the items of data collectible on each, some efficient means of storage and aggregation had to be found. Consequently, a short conversion program R5KTRANS, was written to convert frame use data into format useable by RBase 5000, a popular relational database package. This conversion still has a few bugs, but enough is operational to give an impression of its eventual capabilities.

Additional utilities, LISTER.COM, which generates line-numbered listings with errors flagged, and VTXDEBUG executes the program in a stepwise fashion to assist in debugging.

4. Connecting developed software to our instructional LAN has not been attempted. The RUNVTX software and a sizeable frame set can fit easily on 360K floppy disks –– a hard disk is not necessary. So individual machines could download to floppy disks and run simultaneously. How dependence by 25 or so PCs on a central server would work, we have yet to determine. We have an additional problem of 8 color graphics generation on all LAN PCs, since only one is equipped with the requisite Plantronics board. PC–Videotex will generate images with the standard color card, but only 4 colors are provided, and they are not the colors encoded on the frame generator. The EGA behaves in the same fashion, except that with an appropriate driver, we may be able to develop 8 color output faithful to the original. With our mix of

standard color cards and EGA on the LAN, we have more work before all problems are solved.

WHAT WAS LEARNED ABOUT INSTRUCTIONAL COMPUTING:

Our final product (or as final as it is at the present stage of development) does resemble well what we had envisioned. Indeed, the fortuitous appearance of PC–Videotex as an IBM product saved us from a very difficult circumstance of having to adapt Canadian Telidon soft ware to AT&T NAPLPS. The FCTCOMXY software was also found "off–the–shelf" having been developed sometime earlier at Columbia University as part of a CAI project. RUNVTX seems to provide a basic set of programming tools and use data sufficient to a start–up effort which we will attempt this spring. Given its clear structure in Turbo– Pascal, other features can be added as needed. For the moment, we will work with this basic tool to discover strengths and limits of the system as it now stands.

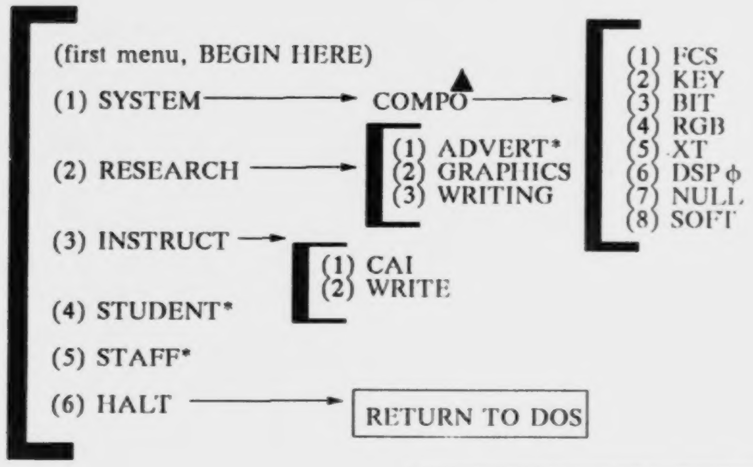
Perhaps the general lesson we had to learn was a compromise of complexity (and the flexibility it entails) and the systems utility to non–expert users. Already, the overhead of setting up the system seems difficult to many, yet with fewer features the system would seem greatly diminished in capacity.More specific lessons concern the use of artifactual data. We soon realized the considerable quality of information this system would generate, moving us to create an interface of Rbase 5000 and SPSS to manage the data and effect its statistical reduction.

Finally, we reinforced a lesson we realized at the onset: the marriage of IBM and AT&T was as difficult as their competitive postures would suggest. We are still troubled by differences between AT&T and IBM in interpretation of the NAPLPS standard, a major bug which defies easy solution until better documentation is forthcoming from both. The AT&T documentation was written to satisfy central switching office engineers. Conversely, the IBM documentation was so pleasantly superficial in critical places that we ended up wishing for more detail. In all, however, we have been pleased at our progress in making a useful tool from this diverse assortment of software and hardware.

EXAMPLES

Attached is a flow chart of the demonstration software, the RUNVTX code which implements it and samples of use data converted to Rbase 5000 format and the output from Rbase 5000 summarizing a session.

DEMONSTRATION PROGRAM FLOWCHART
for Videotext Frame Display



HOME KEY → Begin
PROGRAM KEY → Previous frame
up key → Previous frame
* → Has write field
ϕ → Is an alias
▲ → selected by any key

RUNVTX COMMAND STRUCTURE (CODES FOR IMAGE CONTROL)

AE	add on error	HU	halt on user interrupt
BE	beep on error	IF	if
BU	beep on user interrupt	QU	query (awaits single key user input)
CH	choose	PD	PDP stack return
DI	display frame	RD	read string
GT	go to	RT	return
GS	go sub	PA	pause
HE	halt on error to DOS	TR	trace
HL	halt to DOS	UD	collect data

Demonstration Program

```
UD Y

:BEGIN
DI Begin

:AGAIN1
QU
IF 1
GT SYSTEM
IF 2
GT RESEARC
IF 3
GT INSTRUC
IF 4
GT STUDENT
IF 5
GT STAFF
IF 6
HL
BP
GT AGAIN1


:SYSTEM
DI SYSTEM
QU
IF %UP
GT BEGIN
IF %PGUP
GT BEGIN
IF %UP
GT BEGIN


:COMPO
DI COMPO
:ACAIN2
QU
IF 1
GT FCS
IF 2
GT KEY
IF 3
GT BIT
IF 4
GT RGB
IF 5
GT XT
IF 6
GT DISP
IF 7
GT NULL
IF 8
GT SOFT
IF %HOME
GT BEGIN
IF %PGUP
GT SYSTEM
IF %UP
GT SYSTEM
BP
GT AGAIN2

:FCS
DI FCS
:NEWTRY2
QU
IF %HOME
GT BEGIN
IF %PGUP
GT COMPO
IF %UP
GT COMPO.
BP
GT NEWTRY2
```

```
:KEY
DI KEY
:NEWTRY3
QU
IF %HOME
GT BEGIN
IF %PGUP
GT COMPO
IF %UP
GT COMPO
BP
GT NEWTRY3

:BIT
DI BIT
:NEWTRY4
QU
IF %HOME
GT BEGIN
IF %PGUP
GT COMPO
IF %UP
GT COMPO
BP
GT NEWTRY4

:RGB
DI RGB
:NEWTRY5
QU
IF %HOME
GT BEGIN
IF %PGUP
GT COMPO
IF %UP
GT COMPO
BP
GT NEWTRY5

:XT
DI XT
:NEWTRY6
QU
IF %HOME
GT BEGIN
IF %PGUP
GT COMPO
IF %UP
GT COMPO
BP
GT NEWTRY6

:DISP
DI DISP
:NEWTRY7
QU
IF %HOME
GT BEGIN
IF %PGUP
GT COMPO
IF %UP
GT COMPO
BP
GT NEWTRY7

:NULL
DI NULL
:NEWTRY8
QU
IF %HOME
GT BEGIN
IF %PGUP
GT COMPO
IF %UP
GT COMPO
BP
GT NEWTRY8
```

```
:SOFT
DI SOFT
:NEWTRY9
QU
IF %HOME
GT BEGIN
IF %PGUP
GT COMPO
IF %UP
GT COMPO
BP
GT NEWTRY9

:RESEARC
DI RESEARC
:AGAIN4
QU
IF 1
GT ADVERT
IF 2
GT GRAPHIC
IF 3
GT WRITING
IF %HOME
GT BEGIN
IF %PGUP
GT BEGIN
IF %UP
GT BEGIN
BP
GOTO AGAIN4

:ADVERT
DI ADVERT
RD
GT RESEARC

:GRAPHIC
DI GRAPHIC
:AGAIN5
QU
IF %HOME
GT BEGIN
IF %PGUP
GT RESEARC
IF %UP
GT RESEARC
BP
GT AGAIN5

:WRITING
DI WRITING
:AGAIN6
QU
IF %HOME
GT BEGIN
IF %PGUP
GT RESEARC
IF %UP
GT RESEARC
BP
GT AGAIN6


:INSTRUC,
DI INSTRUC
:AGAIN7
QU
IF 1
GT CAI
IF 2
GT WRITE
IF %HOME
GT BEGIN
IF %PGUP
GT BEGIN
IF %UP
GT BEGIN
BP
GT AGAIN7
```

Demonstration Program Continued

```
:CAI
DI CAI
:AGAIN8
DU
IF %HOME
GT BEGIN
IF %PGUP
GT INSTRUC
IF %UP
GT INSTRUC
BP
GT AGAIN8

:WRITE
DI WRITE
:AGAIN9
DU
IF %HOME
GT BEGIN
IF %PGUP
GT INSTRUC
IF %UP
GT INSTRUC
BP
GT AGAIN9
```

```
:STUDENT
DI STUDENT
RD
GT BEGIN

:STAFF
DI STAFF
RD
GT BEGIN
```

Use Data Converter to ASCII

```
COMPO\4\10\75\3\0\7\\1
FCS\1\0\0\1\0\ \\1
KEY\1\10\10\6\0\ \\1
NULL\1\9\9\4\0\ \\1
STUDENT\1\9\9\11\0\ \ROBERT\1
```

Use Data (ASCII) Converter to R Base 5000

```
FRNAME   USED       DURFRST    DURTOTAL   SEQUENCE   ERRORS     QUERY
-------- ---------- ---------- ---------- ---------- ---------- --------
FCS               1         44         44          1          0
STUDENT           1          9          8          2          0
COMPO             4         10         75          3          0 7
NULL              1          9          9          4          0
KEY               1         10         10          6          0
```

AN ADMINISTRATOR'S VIEW OF AEP PROGRAM OPERATION AND EVALUATION

Presenter: Hellmut Golde *University of Washington*

INTRODUCTION

My purpose today is to talk about the AEP program from the perspective of a university administrator.

In addressing this issue, I will use the OLYMPUS program at the University of Washington as an example. Although the educational environment and the thrust of the AEP program are different in other institutions, evaluation issues are similar. I hope that I will not be perceived as being negative about our program; I am enthusiastic about it, but others on campus need harder evidence than enthusiasm.

UNIVERSITY BACKGROUND

First, I will give you some background about the University of Washington, a large, state-supported, comprehensive urban University. We have well over 20,000 undergraduate students, 7,300 graduate students, and 1,300 students in professional schools; the academic program is organized into 15 schools and colleges. The annual operations budget is of the order of $500 million. Less than one-half of the budget comes from state appropriations and student tuition and fees. In fiscal 1985, the university received well over $200 million in grant and contract awards, placing it among the top 5 US universities in that category.

Over the past 6 years or so, state funding to the university has been steadily decreasing, as measured in constant dollars. Funds to support the educational mission continue to be extremely scarce, including those for academic computing activities. We have been fortunate to acquire a substantial array of computing equipment over the years, to a large extent due to gifts, vendor discounts, research activities, one-time equipment allocations from the state, and the AEP program.

One further comment about the university is worth making: it is a very decentralized organization where decision making rests with schools, colleges and departments. Much of the computing activities and resources are controlled at the unit level, including the budget after it has been allocated. This decentralization has advantages and disadvantages; we believe that the former outweigh the latter.

THE OLYMPUS GRANT

The proposal to IBM limited the extent of the AEP program to the colleges of Arts and Sciences (which includes over three–quarters of the undergraduates) and Engineering. This restriction had the positive effect of focusing the program upon undergraduate education rather than diffusing it throughout a large university. It had the negative effect of dividing the university into participants and non–participants, causing a fair amount of discontent.

This discontent was exacerbated when we had to find the financial resources to support the OLYMPUS program. The State of Washington operates on a biennial budget, and the OLYMPUS program began halfway through a biennium, well after the 1983–85 budget had been distributed to the schools and colleges. The program required "scraping together" sufficient funds to support it adequately for the first year, approximately $1.2 million. This necessitated negotiations with deans, some coercion, a reallocation of some central computing resources, and the total depletion of what little reserves we had.

For the subsequent 2 years, a new biennium, approximately $2.8 million were allocated to the OLYMPUS program, largely from self– generated funds and some reallocation of other computing moneys.

This investment by the university was made because we were convinced that the AEP program would do more toward improving education than some other endeavor; we have strong indications that this belief was justified.

EVALUATION

Let me now turn to the topic of evaluation. The main question the president or academic vice president will ask as we approach the end of the AEP program is "Given the money and other resources the university invested, what is the educational payoff?" Let me pose the question differently: "Could the University of Washington have invested the $4 million of support funds better in some other educational project?"

Before addressing this question, let me state some measurable characteristics of the program.

We will have acquired a mid-sized mainframe computer and over 500 workstations with a combined list price value of $8 million. Many of the workstations are and will remain available to undergraduates in clustered facilities, others are used by faculty in many disciplines. Thus we have greatly increased our hardware resources. It is clear that we could not have acquired and supported these resources without the AEP grant. We now begin to worry whether this new hardware is used effectively – and we do have conflicting indications.

We will have invested a fair amount of manpower in educational software development. The extent to which this software will be used routinely by us and other universities remains to be assessed.

We are introducing computing into many nontraditional disciplines. The AEP program has greatly accelerated this inevitable trend.

These are positive results from the perspective of an educator or administrator who believes that excellent computing facilities are essential to higher education. It does not address the question posed above: "Could we have invested $4 million better in other educational activities, e.g., other equipment, office support, sports facilities, or a new educational program?"

I believe this question can never be unequivocally answered because we cannot perform a controlled experiment. However, we can do quite well by an honest assessment of the program. The following questions are among those that need to be addressed to answer the basic question posed above:

(1) What other educational opportunities have been lost due to the investment of $4 million in the OLYMPUS program?

It is difficult to provide an answer since we do not know exactly what we would have done with the money. We may have bought a certain (but smaller) number of workstations, we may have augmented our central mainframe facilities, we may have hired additional teaching assistants in some departments, we may have done any number of other things. We do not keep a university-wide list of priorities and fund them top down as budget is available.

It is most likely that the $4 million would have been allocated to a number of activities as incremental budgets, allowing us to do those things better.

(2) What changes occurred in the educational process?

Enhancing undergraduate education is the principal objective of the program. An honest, albeit qualitative, determination of how the educational process was changed will go a long way in determining the value of the program. This assessment must involve faculty and students and document details through examples.

We have good evidence, and you have seen some of it at this and the previous AEP conference, that we are bringing change to education. The real payoff will occur when we move from demonstration projects to routine use of the hardware and software.

(3) What additional computing facilities have been provided to students through the OLYMPUS program?

On our campus we can unequivocally say that the computing environment has been greatly enhanced by the AEP program. We still need to determine the extent to which this equipment is used effectively or will be after the OLYMPUS program formally terminates.

(4) In which areas of the university was the program successful and in which areas was it a failure or only marginally successful?

While the answer to this question may not effect the assessment of the university–wide effectiveness of the program, it will provide us with valuable information about discipline–related differences in effectiveness. We must be sure to understand why such differences exist and what we can learn from them. Faculty enthusiasm, existing computer literacy, a "critical mass" in personnel, can contribute to excellent performance in an academic unit, while insufficient support, bad management, exorbitant ambitions, and the subject matter itself can have the opposite effect.

(5) What are the implications for the future?

Clearly, we will continue to support the equipment after the AEP grant terminates. Do we also need to provide support for software projects that were not completed? And will the OLYMPUS project accelerate the demand for more and more equipment?

There are undoubtedly other questions whose answer will materially contribute to an evaluation as viewed from the president's or academic vice president's office. The principal message I would like to leave with you is the following: It is easy for computer professionals to enthusiastically endorse the AEP program because of its equipment impact on the campus. University administrators must be provided with sufficient quantitative data to evaluate the program against other educational investments. This is not easy to do, but must be attempted.

CONCERNS

Let me conclude with a brief comment on two peripheral concerns about the AEP program.

My biggest concern is with the rate of introduction of all this additional computing power into the instructional program and its potential inefficient use, at least initially. It is possible that some units were "overwhelmed" by too much too fast. Is there an optimal rate of introduction of computing into an educational program?

My second concern, related to the first, is whether the speed of software development will be sufficient to make effective educational use of the equipment. There is a danger of having a number of incomplete software projects, unless we continue to support this activity beyond the end of the AEP program.

These concerns and others need to be addressed as part of the internal evaluation to provide guidance for future programs of this kind.

BEYOND AEP

Presenter: Norman Mintz *Columbia University*

I want to discuss some of the problems of the growth of computing in universities. However, my view of the place of demand for computing in the university is inherent in my view of the place of academic support services generally, what I call the academic infrastructure, of which computing is only a part. Therefore, let me set the context for my discussion of computing.

The academic infrastructure, for the purposes of this discussion, comprises the resources scholars require to do their work, research and teaching. It includes offices, classrooms, laboratories, libraries, scientific equipment, computers, telephones, duplicating machines, etc. It is casually observable that this infrastructure in any major university has grown more rapidly than the number of faculty or students, and it is arguable that recently this infrastructure has been growing at an increasing rate though I have not tried to ascertain the fact and it is not necessary to my point.

Despite the rapid growth of the infrastructure, one portion or another is frequently held to be inadequate to the needs of the scholarly community. Thus great publicity was given recently to the finding that the scientific instrumentation available to university researchers is, on average, old and obsolete. Real pressure exists in the Physics community to build a new accelerator larger, more powerful and enormously more expensive than anything we have had before. Similarly, plans advance for large scale arrays for Astronomy. Scanning electron microscopes are now common and higher and higher power NMR machines increasingly necessary in the life sciences.

Since more is published every year, libraries must acquire, catalogue and store more material every year (there the rate of growth has been exponential since the time necessary for collections to double has been decreasing). Moreover, library users are demanding easier, faster access not only to the collections held by their own institutions but to materials held anywhere in the world, including proprietary data bases available only for fees.

Finally, I do not need to tell this audience about the growth of computer power, use, numbers or interconnections on university campuses.

I am leaving out the costs of administration, which has also increased as the need to administer the growing infrastructure has increased, and for other reasons not central to my argument here.

Taken together, the costs of providing the demanded academic infrastructure have been growing, absorbing a greater and greater proportion of the university budget. The cost of faculty as a proportion of university budgets has been decreasing. At the same time, teaching loads have been declining over the last few generations, although, for all practical purposes they have been flat in the recent past. Taken together, all these factors have increased enormously the cost of putting a faculty member in front of a class of students.

This means that productivity in teaching, defined only as students taught per faculty member, and holding class size constant, has declined. Of course, modifications to teaching patterns including larger class sizes in some institutions and increased use of graduate students as instructors has modified the size of this decline, but, I am certain, has nowhere reversed the decline.

In my own institution we have made little use of these ameliorating mechanisms, and the decline in faculty productivity, as I have defined it, is substantial. Note that this definition of productivity does not give any weight to what is taught, how well it is taught or what or how well students learn. I have omitted these factors consciously, because of the structure of university revenues,as I will explain in a minute.

Now consider the sources of the university's funds to support the university's costs, and here I am speaking from the point of view of a private, not a state supported, institution. First there is endowment. While endowment has been growing at most institutions, both from new gifts and from income and appreciation, it has been growing at nothing like rate of university costs, and endowment income has, at most institutions, provided a decreasing proportion of total university expenditures over time. Current gifts, historically, have supported new university initiatives rather than the core of ongoing activities and examples of cases where the acceptance of gifts has increased rather than reduced university operating costs are legion. Universities are dealing with this problem by directing their fund raising effort more and more to raising substitutional funds, those gifts which substitute for funds which the university is already spending or expects to spend. Research contracts and grants, whether governmental or private, support only the direct costs of research and, OMB to the contrary, only part of the indirect costs of that research. As we have seen, that support, while substantial, has still left universities unequipped to do some kinds of scientific research. All the above sources of funds are relatively non–discretionary; the university can control their size or timing to only a limited extent.

Finally there is tuition. Tuition is paid by the point or course and is invariant with the quality of the course, the size of the course or the amount the student learns. This is where the productivity issue bites. The increased size, sophistication and quality of the academic infrastructure unquestionably has improved the quality and quantity of research in all fields. It has changed the nature of research in all fields. (Years ago, a humanist could make a career out of building a concordance; today a typist and a computer can do the same work in little more time than it takes to type the text.)

The nature of scholarship has changed, most think for the better. The quality of instruction has improved as well. Advanced materials are introduced earlier into the curriculum. Students routinely learn material that no one knew a few years ago – research unambiguously improves instruction – and they learn it better than their predecessors, thanks to the existence of various technological improvements in the teaching process. But none of that directly affects university income. Universities are paid according to the number of students taught. Since the real cost per student taught has been secularly increasing, other things equal, inflation adjusted tuition charges should be increasing as well. Of course, they have been recently, following a period of high inflation during which real tuition rates decreased. In the long run, however, the real cost to students and their families of tuition charges at universities has not increased, in part due to increasing financial aid from government and other sources, although the real income to universities from tuition has increased somewhat as real tuition rates have increased. How does all of this relate to computing?

Well, in the last few years the most explosive growth in the academic infrastructure has been in the area of computing. At the same time, the financial aid policies which cushioned for students the real rise in tuition are being reversed in Washington. So we are facing a sharply accelerated increase in costs which is unlikely to be able to be met, in private institutions, by commensurate increases in the major controllable revenue item, tuition. This problem reduces in most discussions to the question: How do universities pay for the enormous growth of computing demand on campus?

I wish I had a concrete answer to this question. What I have at the moment are partial, ad hoc answers which divide into three parts: fund raising, constraining the growth of computing, and resetting of university priorities.

Fund raising and constraining growth in some ways go hand in hand. We at Columbia, like all of you, have been beneficiaries of IBM's generosity in many ways and particularly in the AEP project. Although we were aware that the costs of using, maintaining and eventually replacing computing equipment were substantial, we did not recognize the full scope of the project until we began to budget our AEP project.

Taking account of the costs of housing, maintaining, and technically supporting the AEP computing equipment with non–IBM software packages, telecommunications and consulting, we have found that the 5–year cumulative cost to the university will be approximately equal to the initial retail value of the equipment given us. In other words, when we accept a million dollars worth of equipment, we implicitly commit ourselves to increased annual operating expenses of approximately $200,000 per year for the estimated 5–year life–time of the equipment. These numbers, and I want to emphasize this, do not include the cost of faculty time needed to prepare the programs, etc. necessary for the use of the equipment in teaching or for the eventual replacement cost of the equipment at the end of its five–year life.

Thus we have two problems that must be addressed: the operating costs of the donated equipment and the cost of eventual replacement.

It is not realistic to assume that current donors will solve either problem. No one is contributing sufficient funds to meet fully our current operating costs and we certainly cannot depend on future AEP–like grants to our schools.

Since we are incurring substantial costs now and expect to incur them for the foreseeable future, we are trying to ensure that the AEP project and other projects based on gift equipment are integrated fully into the fabric of the institution and contribute to our central instructional and research purposes. We have refused gifts which are not central to the university's goals, which are not substitutional at least in the sense of defraying costs which we would or would want to incur anyway. That is, we have tried to constrain computer growth, even from gifts, to purposes coincident with the university's.

The resetting of priorities is an even more difficult issue. Universities tend to be very conservative and glacially slow in phasing out old programs in favor of new ones. In part, this difficulty in setting new priorities and acting on them stems from the multidisciplinary nature of universities in which no individual discipline is regarded as intellectually superior to another. In part the difficulty also stems from our inability to measure and gain a financial return from improved quality as a business firm would. Unlike a bank which pays for its computers through reduced transaction costs, we simply cannot pay for computers by reducing intellectual transaction costs.

We have tried to deal partially with this issue by putting the burden of operational support for computing on the user schools and departments and hence to force them to make choices between computer support expenses and other departmental costs. The support of the telecommunications infrastructure, on the other hand, has been recognized to be a central administrative responsibility. Putting the operational burden on the individual user departments only makes good sense if there are, in fact, sources of available funds within departmental control or if the department has its own fund raising capability. Too often, this is not the case. Nevertheless, by putting the burden of setting priorities on the department or school, the decision process is funneled through the correct academic decision making process.

Replacement of equipment after five years is still an unsolved problem. We would like to think that the declining cost of computer equipment will make this a trivial problem. Unfortunately, I strongly suspect that the appetite of our faculty and students for computer power will grow even more rapidly than prices will decline. Therefore, I look for relatively little relief as a result of falling per unit cost of computation.

Since general university income is unlikely to provide sufficient funds for a satisfactory replacement program even if new spending priorities are established, I think we have our work cut out for us.

Our problem, ultimately will prove to be whether or not universities can tolerate politically and substantively a rapid enlargement of the infrastructure, particularly the computing infrastructure, at the expense of faculty lines and other support services. Will university departments constrain the breadth of coverage of their fields to support the areas they do cover more intensively? Will universities exploit their comparative advantages to an even greater extent than they now do? Finally, and not trivially, will the new technologies generate increases in the number of tuition paying students per faculty member?

With careful planning, and fund raising, and with the assistance of suppliers and legislators, most of us will make it, but some won't – many schools not at this conference won't. If computing proves to be only partly as important as I think it will be for education, the dissemination of knowledge and for society, we will have an even more obviously multi–tiered educational establishment. But that is a subject for another conference.

CMU TUTOR: AN INTEGRATED PROGRAMMING ENVIRONMENT FOR ADVANCED WORKSTATIONS

Presenter: **Bruce Arne Sherwood** *Carnegie–Mellon University*

Coauthors: **Bruce Arne Sherwood** *Carnegie–Mellon University*
Judith N. Sherwood *Carnegie–Mellon University*

The great potential for educational use of powerful advanced–function workstations is currently limited not only by cost, but also by the lack of a congenial programming environment for educators. For generality and universality, the Unix operating system has been chosen as the software foundation for such workstations [Crecine 1986]. However, the Unix environment and its computer languages were designed for other purposes and do not give much direct support for the major tasks of educational programming (display and interaction). This makes it difficult for non–expert programmers to create educational materials.

To address this issue, an integrated programming environment called CMU Tutor has been developed in the Center for Design of Educational Computing at Carnegie–Mellon University. CMU Tutor makes it much easier to exploit the power of advanced–function workstations, especially within the window–oriented Andrew system developed by the Information Technology Center at Carnegie–Mellon in a joint IBM–CMU project [Morris et. al. 1986]. CMU Tutor enables those with limited programming experience to develop interactive applications, especially educational applications, for workstations without having to depend completely on the assistance of professional programmers.

On university campuses Unix has been the software system preferred by computer scientists, due to the rich set of computer science tools which have been created by and for Unix users. Unix itself is written in the computer language C, so Unix tools tend to be oriented toward support of C programs. In the hands of skilled system programmers, C is an excellent tool for writing system software. It is natural to attempt to write educational applications for workstations in C. However, C has significant disadvantages for this purpose. Many university faculty members have significant programming skills but are not expert professional programmers. In our own C programming and in that of colleagues we have seen a lot of time lost on picky details. Compiling and linking typical educational programs written in C currently takes several minutes, which seriously impedes rapid development. A large part of educational programming consists of displaying text and graphics in complex ways. Like most computer languages, C gives little or no direct support for such displays. For example, C supports only standard ASCII characters and will not permit italic, bold, or foreign–language characters in a source program. If such text is to be displayed, various inconvenient and indirect subterfuges must be employed.

In contrast, the CMU Tutor language and its programming environment are much more hospitable to non–expert programmers. In comparison with standard languages, the CMU Tutor language is particularly strong in the display of text and graphics and in analyzing user inputs. Incremental compilation combines the revision speed of interpreted languages with the execution speed of compiled languages. Detailed error diagnostics are provided. Rich text and graphics facilities are available in an easy–to–use form. A novel display editor generates CMU Tutor source code in a direct and transparent manner. Powerful tools are available for analyzing keyset and mouse inputs. Pop–up menus are easily created. Calculational features are similar to those available in standard languages. An unusual on–line reference manual offers instant execution of documented language features. In the remainder of the paper these features of the language and its environment will be described in detail.

CMU Tutor is incrementally compiled. That is, only those program segments which have been changed are automatically recompiled. As a result, the developer sees the effect immediately after making a change in the source code. Since the source code is compiled rather than interpreted, execution speed is quite fast. Upon entry to a program written in CMU Tutor, those procedures needed immediately are compiled first. Whenever there is a wait for user input, more of the program is compiled in the background, interrupted if necessary for on–demand compilation when the user requires execution of a section not already compiled. This overlapping of execution and compilation insures that there is hardly any observable delay due to compilation. Since only source code is kept in mass storage, not the compiled binary, there is no need to recompile CMU Tutor programs for different kinds of workstations, unlike C programs which require separate binaries to run on different machines. (Although other languages are available on the Unix–based systems for which CMU Tutor was developed, C is by far the best supported and most widely used language.)

Compile–time and execution–time error diagnostics are very specific. When an error is detected, a detailed error message is displayed. The source code is automatically scrolled to display the statement containing the error, and the editing caret is placed in the statement at the character location where the error was detected. This is in striking contrast to C, in which compile–time diagnostics can be extremely vague, and execution–time diagnostics consist of a core dump to be studied with a debugger.

All special text forms supported by the Andrew editor (italics, bold, large, small, centered, etc.) appear as such in the source code, eliminating the need for complex text output commands. For example, the usual mouse–oriented facilities may be used to make some portion of a paragraph be italic. When the program is executed, those words are displayed in italic, without any special commands. Execution–time position and margin controls determine a rectangle within which a text statement is displayed.

Graphics capabilities include labeled graphs, rotatable displays, pattern–filled polygons and disks, and animation of icons. All the major graphics abilities of workstations are accessible through the CMU Tutor language. Automatic scaling to arbitrary window dimensions is supported with options to scale x, scale y, preserve aspect ratio, and scale text.

A novel graphics editor directly generates CMU Tutor source code without the need for a separate command language for creating and editing displays. Figure 1 shows two views of the programming context, before and after adjusting the corner of the triangle. In each view the source code is at the top and the corresponding display is at the bottom. Across the middle are lists of the most commonly–used commands, and these commands can be entered into the source code by pointing at them with the mouse. Clicking the mouse in the display area causes an x–y coordinate pair to be entered into the source code, followed by recompilation and execution of the current unit, which is very fast. The source code for the graphics shown in the figure was generated without touching the keyboard. Note the rich text capabilities, including centering. (Comments at the ends of source code statements in the example are introduced by $$.)

The right-hand part of Figure 1 shows how existing display routines can be easily modified. The mouse is used to select an x-y coordinate pair in the source code (part of the "fill" statement in the example), as though one were preparing to cut it out or change it to italics. This selected coordinate pair can be visually adjusted by clicking the mouse in the display area, which changes the selected coordinate pair and triggers recompilation and execution to create the altered display. The "fine" command in the example defines a virtual fine-grid coordinate system, and the arguments of the "rescale" command specify that x and y should rescale to fill the window, without preserving aspect ratio, but with scaling of text. The graphics editor supports the device-independent coordinate scheme by unscaling pixel-based mouse clicks into appropriate fine-grid positions.

CMU Tutor handles input from both keyset and mouse. Convenient analysis routines are provided for word, sentence, numerical, and algebraic inputs. Sentence analysis includes checks for spelling and word order. Answer "markup" is supported: unanticipated words and miscapitalized letters are shown in bold-faced, misspelled words are shown in italics, etc. Mathematical input can be evaluated at high speed, making it easy to create function plotter programs. Implied multiplication (e.g. "3y") is permitted, so input expressions can be written using normal mathematical notation. Both mouse clicks and mouse movements can be processed. Pop-up menus are easily created and managed.

The usual calculational structures are provided, including if-elseif-else, case, and loop-reloop-outloop (which provides a unified form for iterative, while, and until loops). Case instances need not be constants. Subroutine arguments can be passed by value or by address. Local variables permit recursion. The logical expression "A < B < C" is handled correctly (something few languages do). In the expression "n>0 & J(n)=3", the array reference J(n) is not attempted if n is not greater than 0 (this conditional evaluation of expressions is also available in C).

A common syntax exists for conditional commands. The arguments of any command (other than those defining control structures, such as if) can be selected by the value of an expression. This provides a compact one-line case statement.

An important component of the CMU Tutor programming environment is its powerful on-line reference manual. Language features can be accessed either through hierarchical indices or through names of commands in the language. In either case selections are made simply by pointing with a mouse. Language features are illustrated in context by sample routines. A unique feature is the ability to execute these samples immediately by using the mouse to copy them into the programming window. These sample routines can act as nuclei for further elaboration by the programmer. A printed version of this reference manual is being prepared.

The advanced-function workstation provides an excellent environment for program developers. However, until these workstations come into wider use, the ability to execute CMU Tutor programs on other microcomputers would be desirable. Work is underway at Carnegie-Mellon and at Brown University toward possible delivery of CMU Tutor programs on popular microcomputers (assuming these programs have first been created on workstations).

Although it is already being used for many purposes, CMU Tutor is not complete. The biggest gaps at present are the lack of file input and output (other than referencing fonts and icons) and the lack of rich data types (only floating point variables and one-dimensional arrays are currently available). Closing these gaps is the highest development priority for the immediate future. Plans include offering the kinds of data types available in C or Pascal, including dynamic storage allocation. At present, arithmetic expressions are only semi-compiled, but for even greater speed the compiler should generate actual machine code for whatever workstation is doing the execution. Also planned is the ability to call routines written in C or other workstation-supported languages. For example, a large LISP language-analysis routine might benefit from a user interface written in CMU Tutor. Another high-priority area is course delivery aspects such as routing students among existing materials, generating reports to instructors, etc.

An example of an educational program written in CMU Tutor is "quantumwell" by Bruce Sherwood. The mouse is used to specify a potential well in which an electron is trapped. In Figure 2 the mouse is used to choose a trial energy level, and the quantum-mechanical equations are evaluated left-to-right to plot the corresponding wave function. This program depends on the display, input analysis, and calculational strengths of CMU Tutor. Creating the program was greatly facilitated by the ability to test changes rapidly, thanks to incremental compilation.

CMU Tutor is based on the MicroTutor language developed at the Computer-based Education Research Laboratory at the University of Illinois (the PLATO project). It incorporates MicroTutor's important constructs for interactive educational programming [Stifle, Smith, & Andersen 1979; Sherwood & Sherwood 1985]. MicroTutor is basically an updated and machine-independent descendent of the TUTOR language [Sherwood 1977], whose development was initiated by Paul Tenczar in 1967. The first version of MicroTutor was created by David Andersen in 1977. Similar languages include Digital Equipment Corporation's DAL (Digital Author Language) and Tenczar's own TenCore for the IBM PC. In large part, CMU Tutor is compatible with MicroTutor, but it differs from MicroTutor and other TUTOR-like languages in its extensions for the advanced-function workstation environment, where windowing, mice, and menus impose additional requirements that the earlier languages did not have to meet. The genesis of CMU Tutor is detailed in [Sherwood 1985].

Design and implementation of the basic architecture of CMU Tutor was done by Bruce A. Sherwood, who is associate director of the Center for Design of Educational Computing (CDEC) and the Information Technology Center (ITC), and professor of Physics. Major contributions including much of the graphics and the on-line reference manual were made by Judith N. Sherwood, who is director of educational computing in the Mellon College of Science. Additional contributions to the development of CMU Tutor have been made by CDEC staff members Tom Neuendorffer, Chris Koenigsberg, and David Trowbridge. Jill Larkin, director of CDEC during the period of this development, encouraged the project and helped debug CMU Tutor. ITC staff members Fred Hansen, Tom Peters, and Andrew Appel contributed much valuable advice.

In order to facilitate wide use of CMU Tutor, Carnegie-Mellon University intends to make the C-language source code for CMU Tutor easily available. These routines bear the following statement: "(c) Copyright Carnegie-Mellon University, 1985. May be used and copied freely for educational purposes, but may not be sold without the written consent of Carnegie-Mellon University."

REFERENCES

Crecine, J. P. The next generation of personal computers. Science 231, 935-943 (Feb. 28, 1986).

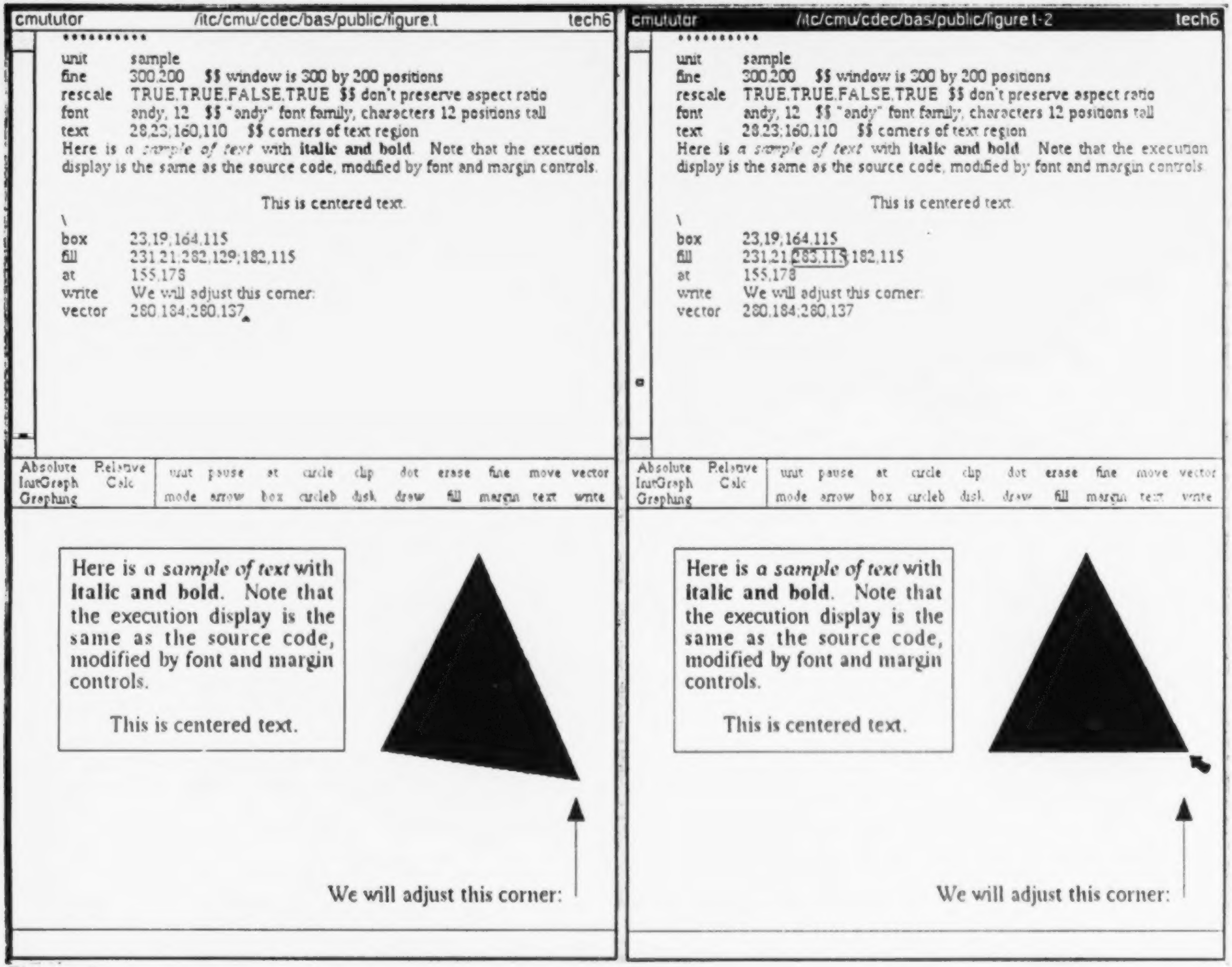
Morris, J. H., Satyanarayanan M., Conner, M. H., Howard, J. H., Rosenthal, D. S. H., and Smith, F. D. Andrew: a distributed personal computing environment. Communications of the ACM 29, 184–201 (March 1986).

Sherwood, B. A. The TUTOR Language. Control Data Education Co., Minneapolis, Minnesota (1977).

Sherwood, B. A., and Sherwood, J. N. The MicroTutor Language. Stipes Publishing Company, 10 Chester Street, Champaign, Illinois 61820 (1985). This book unfortunately is now out of print, but a revised book dealing specifically with CMU Tutor is planned for the near future.

Sherwood, B. A. An integrated authoring environment. Proceedings of the IBM Academic Information Systems University AEP Conference, Alexandria, Virginia, 29–35 (June 1985). Here it is explained that CMU Tutor gets its name from being implemented in C, with MU being the Greek letter for Micro.

Stifle, J., Smith S., G. and Andersen, D. M. Microprocessor delivery of PLATO courseware. Proceedings of the Association for the Development of Computer-based Instructional Systems, San Diego, 1027–1035 (1979).



```
***********
unit      sample
fine      300,200    $$ window is 300 by 200 positions
rescale   TRUE,TRUE,FALSE,TRUE  $$ don't preserve aspect ratio
font      andy, 12   $$ "andy" font family, characters 12 positions tall
text      28,23;160,110    $$ corners of text region
Here is a sample of text with italic and bold.  Note that the execution
display is the same as the source code, modified by font and margin controls.

                    This is centered text.
\
box       23,19;164,115
fill      231,21;282,129;182,115
at        155,178
write     We will adjust this corner:
vector    280,184;280,137
```

Here is *a sample of text* with **italic and bold**. Note that the execution display is the same as the source code, modified by font and margin controls.

This is centered text.

We will adjust this corner:

```
***********
unit      sample
fine      300,200    $$ window is 300 by 200 positions
rescale   TRUE,TRUE,FALSE,TRUE  $$ don't preserve aspect ratio
font      andy, 12   $$ "andy" font family, characters 12 positions tall
text      28,23;160,110    $$ corners of text region
Here is a sample of text with italic and bold.  Note that the execution
display is the same as the source code, modified by font and margin controls.

                    This is centered text.
\
box       23,19;164,115
fill      231,21;283,115;182,115
at        155,178
write     We will adjust this corner:
vector    280,184;280,137
```

Here is *a sample of text* with **italic and bold**. Note that the execution display is the same as the source code, modified by font and margin controls.

This is centered text.

We will adjust this corner:

Figure 1.
A novel graphics editor generates CMU Tutor code. On the right, an adjustment is being made to the display. The selected region of the source code is modified simply by pointing at the screen to indicate the new coordinate.

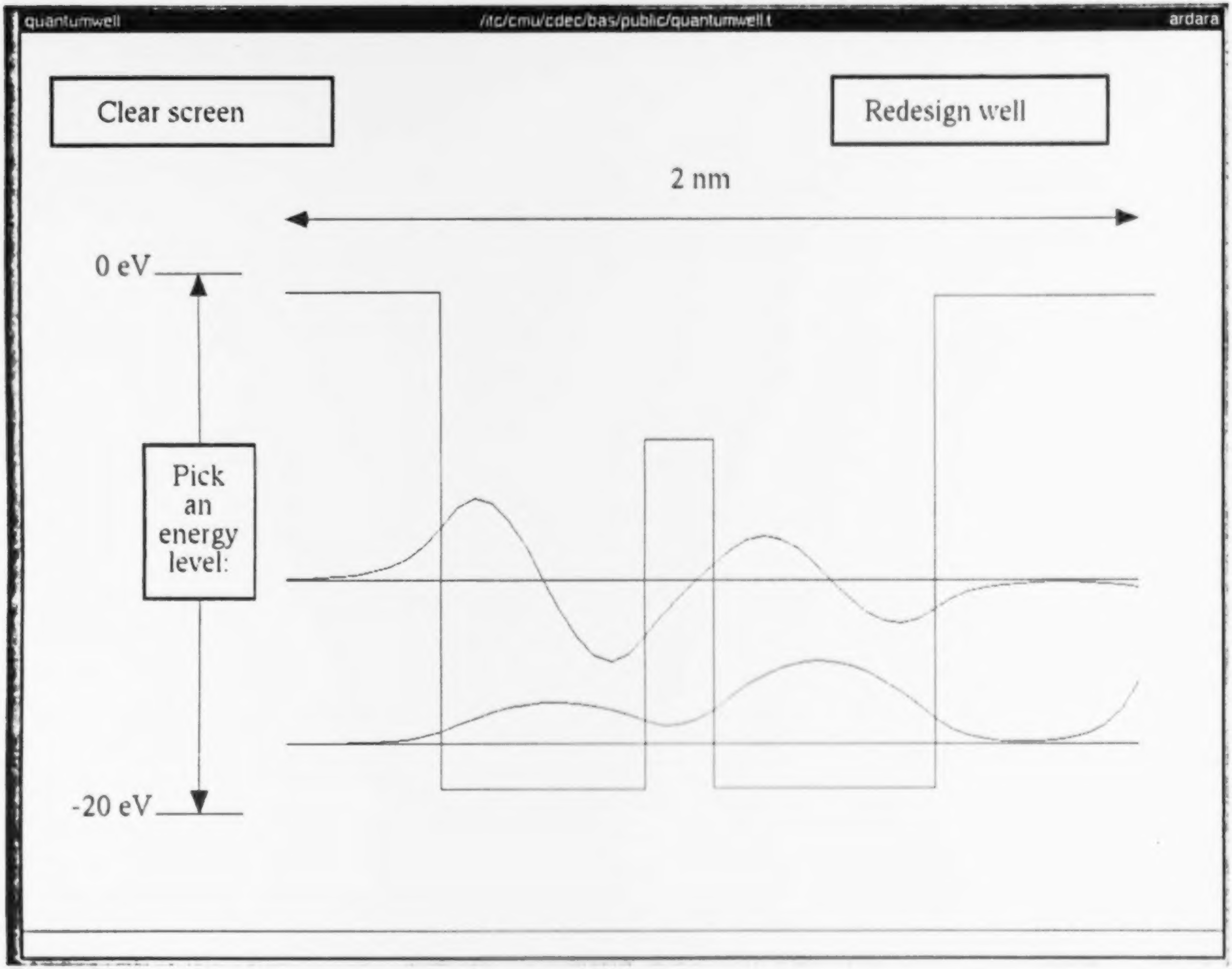


Figure 2.
An example of an educational physics application written in CMU Tutor. After defining a potential well with the mouse, the user selects a trial energy level. The quantum-mechanical equations are evaluated and the trial wave function plotted.

COMPUTER AUGMENTATION OF FLIGHT SIMULATOR TRAINING: PROJECT OVERVIEW

Presenter:	**M. H. Weller**	*University of Illinois*
Coauthors:	**M. H. Weller**	*University of Illinois*
	C. D. Wickens	*University of Illinois*
	J. M. Flach	*University of Illinois*
	T. Davis	*University of Illinois*
	A. F. Stokes	*University of Illinois*
	W. C. Entwistle	*University of Illinois*
	G. Lintern	*University of Illinois*
	E. Billman	*University of Illinois*
	R. J. Braune	*University of Illinois*
	A. Mane	*University of Illinois*

ABSTRACT

Project objectives are to standardize instruction; incorporate realistic environmental cues (e.g., radio communications and "outside-the-cockpit" visual displays); augment performance feedback; and automate aspects of student performance monitoring for the flight simulator portion of the Institute of Aviation's introductory flight course. The networking of two Illinois Micro Aviation Computer (ILLIMAC) digital flight simulators with three IBM PC-AT's provides instructor control and monitoring of both simulators from a single PC-AT. Students in each simulator receive instructions, performance feedback, and supplementary graphic flight profile information from an IBM Professional Graphics color display mounted on the simulator.

Two aspects of the project, relating to visual cues for landing and radio communications, are being expanded for part-task training on an IBM PC-AT that is not part of the networked configuration. The graphics capability of the PC-AT is being used to create real-time display for training students to land an aircraft. Students will control the position of a simulated aircraft by means of a two-axis joystick and an auditory "warning" tone will be generated upon deviation from the proper glideslope.

For training radio communications skills and airport procedures, a digital audio recorder/playback device will be interfaced with the IBM PC–AT. This device allows for the presentation of a sample radio transmissions recorded in–flight. The transmissions will be accessed both in tutorial components to train the fundamentals of radio communications and during more interactive units to exercise student communication skills.

All components of the project are discussed in terms of task organization, progress to date, and evaluation plans.

INTRODUCTION

The skill of flying an aircraft is demanding, complex, and in many respects, unique. The pilot must simultaneously keep the plane from falling out of the sky, move it from one point in space to another, and communicate with air traffic control. To do this, information from many instruments and form the dynamic world outside the cockpit must first be understood and integrated, and then translated into a coordinated series of performances, often using both hands and feet. The complexity of this skill, and its inherent dangers are sometimes overwhelming to the novice, and have encouraged us to incorporate dynamic graphics into the simulator portion of flight training in the Institute of Aviation's introductory flight course.

PROJECT CONTEXT: INSTITUTE OF AVIATION

The Institute of Aviation, directed by Dr. Henry L. Taylor, is an academic unit of the University of Illinois at Urbana–Champaign and is made up of subunits devoted to instruction, research, and service. The instructional units, the Pilot Training Department and the Aircraft Systems Department, accept students with a wide range of interests in aviation. Students in the pilot training curricula typically obtain at least a private pilot's license, and may continue their instruction in the areas of instrument flight, multi–engine, commercial, airline transport, and flight instruction.

The research mission of the Institute is carried out primarily under the auspices of the Aviation Research Laboratory (ARL). ARL faculty are multi–disciplinary, typically holding joint appointments within the Institute or with one or more campus departments such as Psychology and/or Mechanical and Industrial Engineering.

Our EXCEL project is a joint undertaking of the Pilot Training Department and ARL. Each of the units brings to the effort staff with vital expertise: Pilot Training offers aviation content specialists and master teachers, while ARL has faculty and staff knowledgeable in instructional training technologies as well as hardware and software specialists.

OVERVIEW OF PROJECT GOALS

The focus of our project is the beginning flight course, Aviation 101 and, more specifically, the training provided in the flight section prior to the student's first lesson in the aircraft. Our efforts can be summarized in terms of four main objectives:

1. Standardize instructional content for this portion of the curriculum.
2. Automate appropriate aspects of instruction.
3. Augment and enhance student performance feedback in the simulator.
4. Introduce part–task training on visual aspects of landing skills.

Understanding why these are viewed as important objectives requires a brief discussion of the Aviation 101 curriculum.

AVIATION 101 CURRICULUM

Aviation 101 is the first course in a two–course sequence leading to FAA Private Pilot Certification. On an average, approximately 90 students enroll in the course in the Fall semester, with an additional 20 students taking the class in the Spring. Students enroll in both a ground school section and a flight section, each taught on an individual basis. During the academic year, a full–time flight instructor is responsible for the flight instruction of 12 students per semester. With each student in Aviation 101 required to log six hours in the flight simulator and 27 hours in the aircraft, this represents a considerable load in terms of instructor contact hours.

The flight instructor not only teaches the student in the simulator and in the aircraft, but also spends a great deal of time tutoring the student on key contact areas. The quantity and content of this tutoring varies according to both the student and the instructor, but typically includes at least four hours of instruction before simulator training begins. During this time the instructor covers such topics as the instruments used to control the aircraft, radio communications at Willard Airport, and checklist procedures to follow in flight. The student then generally spends three hours in the flight simulator practicing various maneuvers. In addition, the flight instructor provides practice on radio communications procedures by playing the role of various air traffic controllers and requiring the student to initiate communications and respond to instructions.The remaining three hours of simulator time are completed at various points during the semester to provide additional practice on specific areas. Tutoring by the instructor also continues as needed throughout the semester.

Another aspect of the curriculum relevant to the project concerns the capabilities of the flight simulators available for training. Aviation 101 students are currently trained in a motion-based Link General Aviation Trainer (GAT) connected to a plotting device designed to trace the student's performance along an assigned flight path. Although these plottings are useful to both instructor and student at the end of a lesson to review performance off-line, student feedback during the flight is limited to the comments made intermittently by the supervising flight instructor, which are made on the basis of subjective evaluations of the student's performance. Finally, although this trainer can provide practice on landings in terms of power settings and instrumentation, it does not present any direct visual cues to help the student learn to land. In other words, the trainer by itself provides no visual representation of the world outside the cockpit. Our goal is to use the unique graphics capabilities of the IBM PC-AT to provide the visual representation and on-line visual feedback of flight information to the student. To do this, we will be networking the IBM PC-AT to the Illinois Micro Aviation Computer (ILLIMAC) flight simulator, which was designed and developed at the Institute of Aviation.

The EXCEL project provided the Institute with the opportunity to examine the pre-flight portion of the Aviation 101 curriculum from the perspective of how the use of computer technology could significantly enhance the instruction while making better use of flight instructor time.

Keeping in mind the broad objectives of standardizing and automating aspects of instruction, providing better performance feedback in the simulator, and introducing part-task training on landing skills, we have developed a three-phase approach for our first year of development.

PHASE 1: RADIO COMMUNICATIONS AND AIRPORT PROCEDURES

During our development work, it soon became apparent that the individualized instruction currently given on radio communications and airport procedures was suitable for presentation as an independent instructional unit on the IBM PC–AT. Given the difficulty of understanding radio transmissions when the learner is typically first exposed to them––under the stress of flying in the aircraft––we felt that a program of pretraining communication skills would be of particular benefit. We are currently developing a one–hour computer tutorial that should standardize the information presented to students and reduce the time flight instructors currently spend teaching the material.

To program the lesson, we are using TENCORE (Computer Teaching Corporation, Champaign, IL), which is a lesson authoring system developed for the IBM PC. The intent of the program is to familiarize students with the physical aspects of Willard Airport, in terms of location of ramps, taxiways, and runways; with the format (e.g., structure and terminology) of the communications themselves. An audio playback device, called the INSTAVOX (Education & Information Systems, Inc., Champaign, IL), is interfaced with the computer and is used to present both instructional content and recordings of radio transmissions from Air Traffic Control.

Teaching radio communications to student pilots is somewhat analogous to teaching a foreign language. The student must master:

1. Specific terminology and vocabulary (including conventions concerning how numbers and words are pronounced over the radio to facilitate comprehension);

2. Syntactic conventions (a sort of "grammar" that determines the order in which information is stated); and the

3. Cultural context (the responsibilities of the various air traffic controllers and the physical world in which they operate).

This last aspect explains why our radio communications lesson also includes a thorough presentation of the physical dimensions of the airport and the conceptual dimensions of the controller responsibilities––the student must understand this microcosm in order to communicate about it.

Although the general content of the lesson had always been a part of the Aviation 101 curriculum, the presentation of the material had never been systematized. We began with the informal task analysis of what the beginning student needs to know in order to communicate with air traffic controllers. Since most of the flying done by Aviation 101 students occurs immediately around the U. of I. Willard Airport (as opposed to cross–country flights), we further limited our attention to this particular airport, its physical layout, and air control facilities. Our task analysis was somewhat eclectic in format––including such activities as recording radio transmissions in flight, transcribing and discussing them; reviewing instructor handouts currently used to teach the material; and preparing outlines of specific topics to be covered in the lesson.

Through this process we were able to develop an overall content plan for the computer–assisted instruction unit, which consists of four subunits:

1. Air Traffic Control responsibilities (tutorial).
2. Willard Airport Layout (tutorial).
3. Structure of Radio Communications (tutorial using variety of sample transmissions integrated with flight scenarios).
4. Communications practice (using additional sample transmissions within flight scenario contexts).

The next step was to translate the topic outlines for each subunit into a format to be followed by the programmer. We not only had to "flesh out" the outline in terms of specific wording of content, but we also had to indicate how the material was to be presented. We discussed various options, including:

TEXT PRESENTATION VS GRAPHICS

VISUAL PRESENTATION VS AUDIO

Since the task of flying an aircraft is highly visual and spatial in nature and since we want to familiarize students with physical configurations, our lesson requires considerable use of graphics, particularly in the first two subunits. We explored various possibilities in terms of combining graphics and textual presentations on the screen and soon found that we needed most of the screen space for the graphics themselves. At the same time, we were investigating how to provide students with an introduction to the type of radio transmissions they would hear in flight. We soon realized that a speech synthesizing approach would not have an acceptable level of realism to provide adequate training. Just as foreign language teachers use recordings of native speakers, we felt the need to use recordings of communications by air traffic controllers in our lesson. The INSTAVOX, as mentioned earlier, is an audio device that allows their integration with the visual display. In exploring this technology, we decided to use the INSTAVOX to present instructional narrative as well--thereby freeing screen space to meet our graphics needs.

We did not make this decision lightly because there are certain very real risks in presenting instruction auditorily. The major disadvantage centers around the fleeting nature of the spoken word--it does not remain present for the student to "re-hear," as the written word can be "re-read." Figure 1 is a screen print from the lesson illustrating how we have tried to overcome this disadvantage by:

1. Presenting brief textual messages on screen to give redundancy for the audio information;

2. Color coding the graphic representations to coordinate with air traffic control transmissions (Color is represented by shading in Figure 1.); and

3. Incorporating "repeat" options, allowing the student to replay audio messages or to return to a previous "page" or logical unit of instruction.

When the student first reaches this display in the unit, only the basic outline of the airport layout is generated. After a brief computer-controlled pause, instructional narration on the INSTAVOX begins, stating briefly that the student will need to know the location of the ramps, runways, and taxiways at Willard Airport. After another computer-controlled pause, the narrator describes the first airport surface (ramps). The appropriate area on the screen is then filled with a selected color, and a summary message is added to the bottom of the display, The student is then required to press the ENTER key to continue, The same process of audio description, color highlighting, message summarization, and student-controlled continuation is followed for the remaining two elements of the display (i.e., taxiways and runways). The prompts across the bottom of the screen indicate to the student the repeat options available.

The structuring of the repeat options has presented some difficulty in determining just how much information should be repeated at any given time. A student selecting the F7 (repeat) option at any time while viewing the display in Figure 1 will be returned to the beginning point and will hear all three surface descriptions again. Depending on user evaluations, we may decide that the repeat option should be used exclusively for replaying only the most recently presented message.

Another important aspect of the presentation format is how the student progresses through the lesson; that is, whether the material is paced by the program or is fully under the student's control, requiring a key press to continue. We are experimenting with both approaches and hope to arrive at an optimum mix of the two. This is particularly important when using auditory presentation of information. We are intentionally keeping the individual messages very short and using the visual techniques described above to reinforce the narration. Requiring the student to press the ENTER key after such brief messages, however, can cause the presentation to take on a somewhat jerky and tedious aspect.

In prototyping the lesson, prior to programming in TENCORE, we have found IBM Storyboard to be particularly useful. Storyboard allows project members who have skill in creating graphics to sequence their images and incorporate the audio messages without requiring extensive programming skills. The graphic images, created using a graphics tablet and PC Paintbrush, can also be incorporated into the TENCORE environment. This approach has allowed us greater flexibility for initial development and experimentation.

TASK PROGRESS

We have concentrated our efforts to date on the design and development of the first two units of the lesson. These will be pretested on volunteer instructors and students during the 1986 Spring and Summer semesters. The remaining two units, which require extensive recordings of in-flight transmissions, are currently being designed. Since we are still resolving some basic issues related to pacing and the providing of visual redundancy for the auditory messages, we hope to profit from feedback gained through pretesting the first two units prior to enter the programming phase of the latter two. The full lesson should be ready for pretesting during the Fall 1986 semester; an experimental evaluation of all components of the project is scheduled for the Spring 1987 semester.

PHASE 2: AUGMENTING PERFORMANCE FEEDBACK IN THE SIMULATOR: SPATIAL ORIENTATION THROUGH GRAPHICS DISPLAY

By interfacing the PC-AT with the ILLIMAC flight simulator, we will be able to use the high resolution graphic capabilities of the Professional Graphics

Controller and Display to present the two primary displays shown in Figure 2. The display on the left (the "outside-in flight path") indicates the student's performance in terms of maneuvering the craft along an assigned flight path. This display is similar in concept to the plotting device currently in use, but has the clear advantage of presenting the information throughout the lesson, and of providing both top down and side perspectives. On the right of the screen, in a separate window, the student will be shown a representation of his craft's position relative to an artificial horizon. A third window, across the bottom portion of the display, will be used to present textual instructions and feedback to the student.

Figure 3 shows the planned configuration of the networked PC-AT's and ILLIMAC flight simulators. The heart of the network consists of the instructor station, from which one teacher will ultimately monitor the activities of students in two flight simulators. Upon start-up, the on-board digital computer of each flight simulator will transmit status data to the instructor PC-AT via RS232 lines. Through a menu-driven program already in use with the ILLIMAC flight simulator (Taylor et al., 1985), the instructor will be able to alter the default status of the simulator to change, for example, the starting altitude or induce mechanical failures. The new values selected by the instructor are transmitted back to the flight simulators through the same RS232 lines. The program, which currently resides in memory in the simulator's computer and is accessed through a standard terminal, will be rewritten in TENCORE for execution through the IBM PC-AT.

As Figure 3 indicates, the instructor's computer not only receives and transmits data to the flight simulator, but is also networked to a separate IBM PC-AT located beside each simulator. Status data from each simulator will be screened by the instructor's computer and selected components will be retransmitted to the respective simulator's PC-AT. These computers, in turn, will use the data to update the two primary displays showing the student's position along the flight path and relative to the artificial horizon. Analysis of status values, in terms of deviation from the flight path, will trigger various performance messages to the students. The instructor will also be able to intervene directly during the lesson and transmit messages to the student. All flight path data will, of course, be stored and available for post-flight evaluation by the student and instructor.

The instructor's control of each simulator is made possible through access to three displays. The center monitor will provide the instructor with information concerning the instrument readings in the respective flight simulators and, by toggling the screen, will present the available control options. The monitor on either side of the instructor will present in real-time the same display that the student is currently viewing.

Through the networking of the PC-AT with the flight simulator, this project component benefits the student by providing:

1. Continuous online feedback (e.g., deviation from flight path);
2. Orientation relative to an artificial horizon; and
3. Objective flight performance measurements.

A further benefit to the overall training program is a more efficient use of flight instructor time.

TASK PROGRESS

The initial networking of the configuration has been tested with data successfully transmitted between the flight simulator and the PC-AT. Preliminary work for converting the instructor station programming to TENCORE is underway. The basic format of the displays to be presented to the instructor has been outlined. Pretesting of this component will be conducted during the Fall 1986 semester, with a formal evaluation scheduled for the Spring 1987 semester.

PHASE 3: LANDING DISPLAYS

As mentioned previously, the training simulators currently in use do not provide any means for helping the student develop a spatial orientation based on visual information. Since such an orientation is important in developing landing skills, we have targeted this area for inclusion in the overall project. At this point, our goal is to develop a part-task training tool, one that operates independently of the flight simulator—allowing us to isolate the visual aspects of landing from the more complex environment of controlling either a flight simulator or an aircraft. Two major questions face the faculty heading up this component of the project:

1. What information needs to be presented in the screen display for the student to develop landing skills?

2. Assuming that the cues themselves are adequate, can it be demonstrated that an adaptive training schedule using augmented visual feedback will better prepare students for performing landings in the aircraft? (Lintern, 1980)

We cannot overemphasize the partial nature of this training tool. Figure 4 shows a static view of the landing display as it appears when the student is approaching straight to the runway. The display consists of a very simplified horizon line, a runway with centerline, and an approach cue (in this case a vertical pole). The display is dynamic; all aspects change in perspective and dimension as the student controls the position of the "craft" within the display through the use of a joystick. An auditory "warning" tone indicates deviation from the proper glideslope. With the processing power of the PC-AT and the resolution of the Professional Graphics Controller, we are optimistic that we can develop a part-task trainer effective in developing spatial orientation skills vital to landing an aircraft.

TASK PROGRESS

A prototype landing display has been designed and programmed in IBM PASCAL. Pretesting of this component should occur during the Fall 1986 semester, with a formal evaluation to be conducted during the Spring 1987 semester.

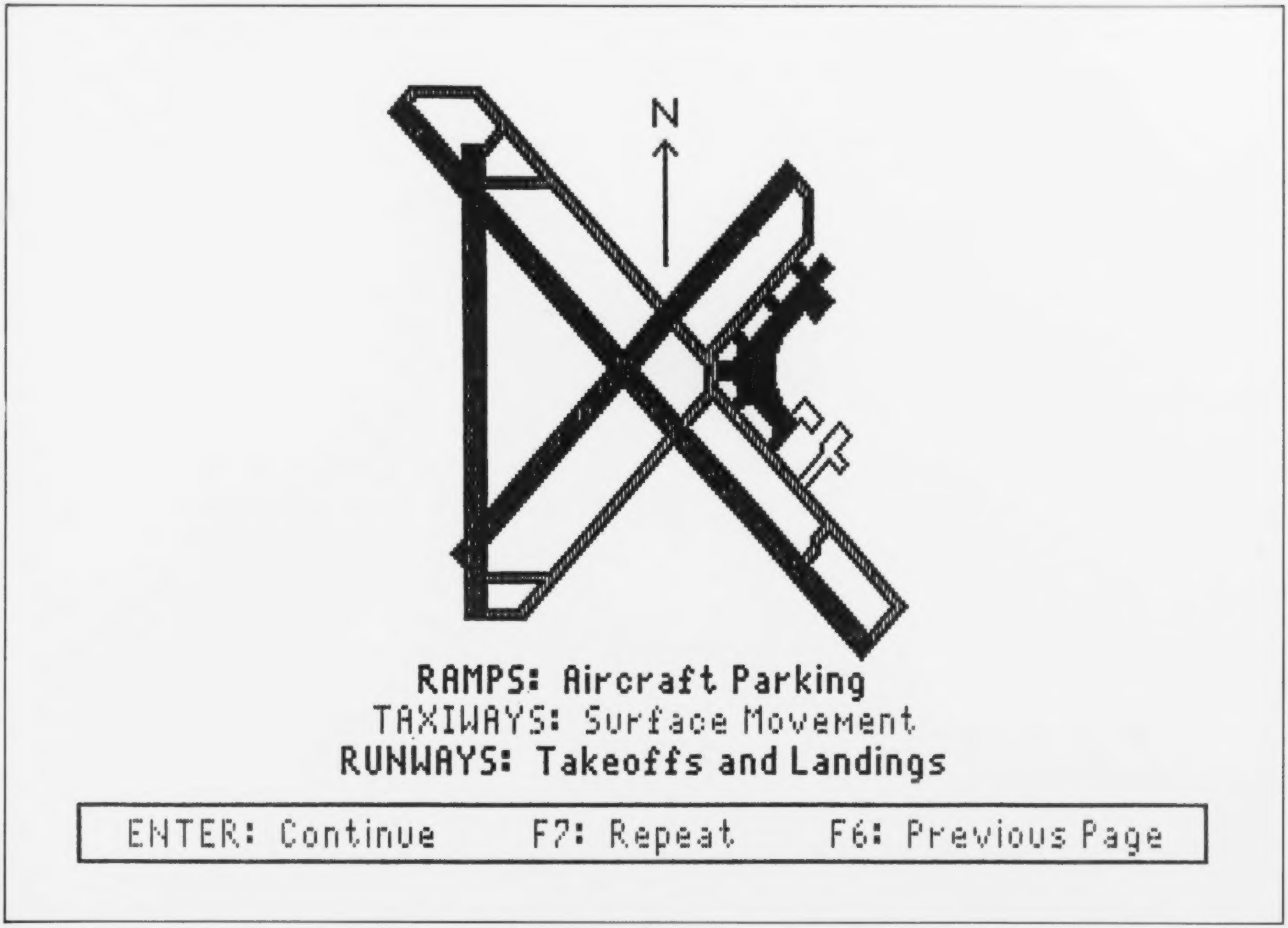
CONCLUSION

The EXCEL project has fostered collaboration among faculty and staff within the Institute of Aviation to apply computer technology to enhance aspects of the introductory flight course. The project, which is an ambitious one involving many personnel, consists of three distinct components designed to provide computer-based training in the areas of radio communications and airport procedures; part-task training on the visual aspects of landing skills; and to provide augmented feedback in the form of graphics displays to students undergoing training in the flight simulator. The three phases of the project are at varying stages of completion; all phases should be ready for formal evaluation by the Spring 1987 semester.

REFERENCES

Taylor, H. L., Hyman, F. C., Todd, R. E., and Hodel, A. S. (1985). Instructor station for the ILLIMAC general aviation instrument training simulator; NAECON–Proceedings of the IEEE National Aerospace and Electronics Conference. 1111–1118. Dayton, OH. May 1985.

Lintern, G. (1980). Transfer of landing skill after training with supplementary visual cues. Human Factors, 22, 81–88.



RAMPS: Aircraft Parking
TAXIWAYS: Surface Movement
RUNWAYS: Takeoffs and Landings

Figure 1. Radio Communication Lesson Sample Display

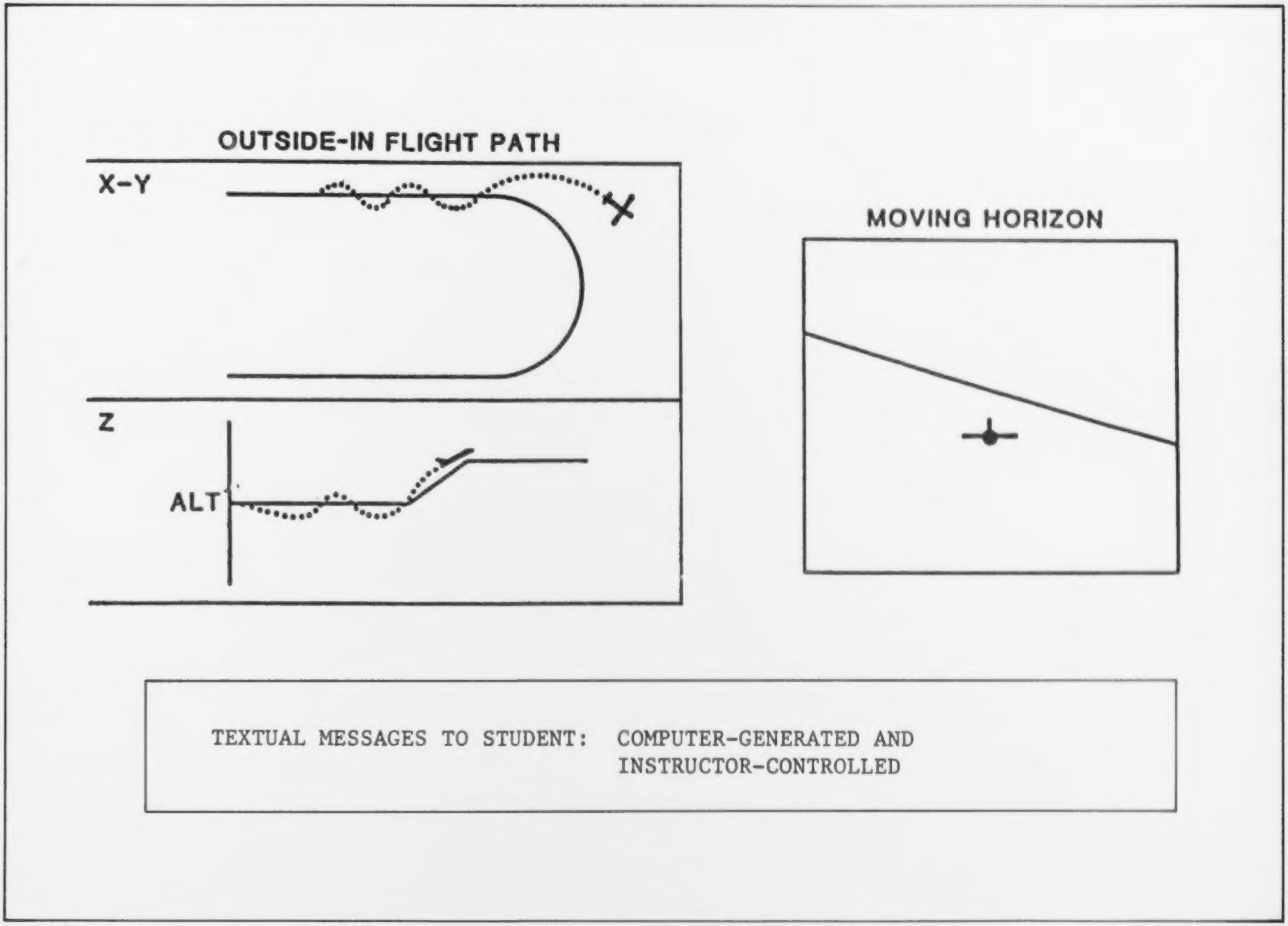

OUTSIDE-IN FLIGHT PATH
X-Y
Z
ALT
MOVING HORIZON
TEXTUAL MESSAGES TO STUDENT: COMPUTER-GENERATED AND INSTRUCTOR-CONTROLLED

Figure 2. Spatial Orientation through IBM Graphics Displays

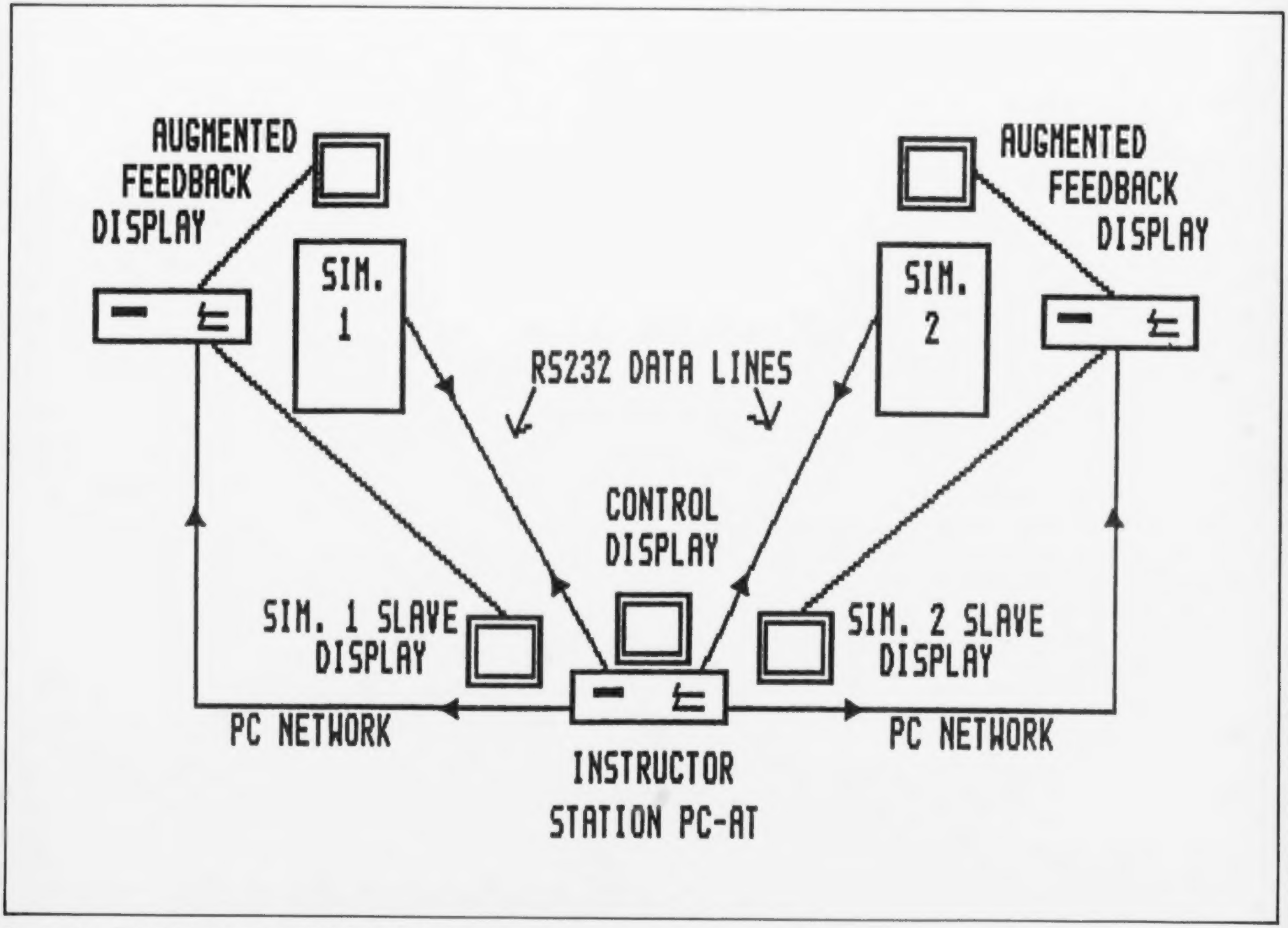


Figure 3. Flight Simulator – IBM PC–AT Network Configuration

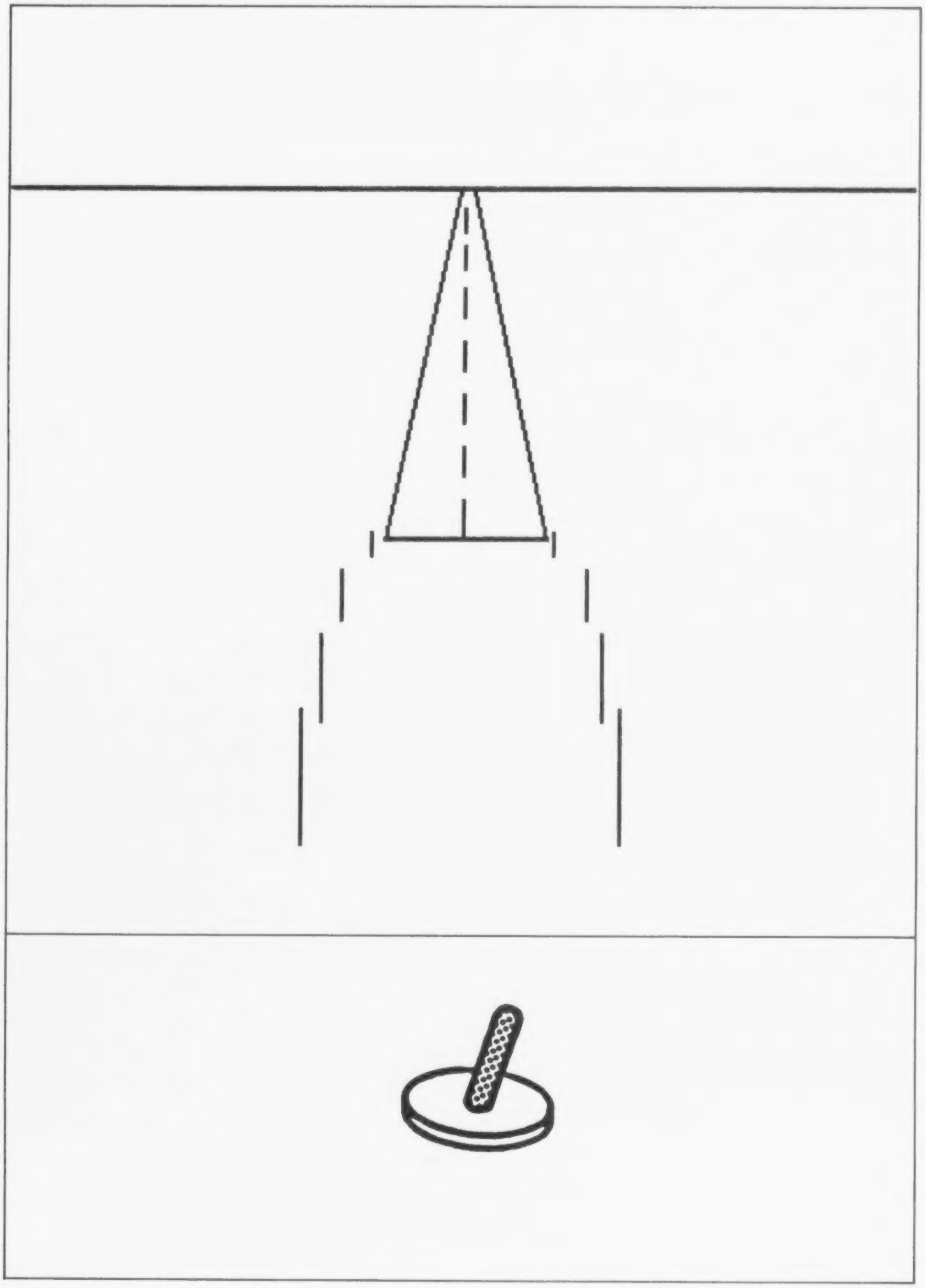


Figure 4. Landing Display for Visual Part–Task Trainer

355

COMPUTERS IN RESIDENCE: SOCIAL AND MANAGERIAL ISSUES

Presenter: Patricia Burton Helm *University of Pennsylvania*

A. THE NATURE OF THE FACILITY AND ITS GOVERNANCE

The University of Pennsylvania's Threshold grant has provided Hill College House with a computer classroom equipped with 14 IBM PCs and two XTs, each with an individual dot–matrix printer. This in–house classroom provides a meeting space and after–hours computer facility for 560 undergraduates, predominately freshmen, who are enrolled in all four undergraduate schools, that is, Arts and Sciences, Engineering, the Wharton Business School, and Nursing. Our objective was to set up courses for Hill students, to be held in the residence itself in Hill House, in a classroom where students and faculty could meet during the day and where students could use the computers individually in the evening.

The Hill House Computer courses and the facility which supports them are a natural outgrowth of the philosophy of the House. During the last fifteen years, resident and affiliated faculty, assisted by our own resident staff of graduate and professional students, have tried to encourage students' academic interests within the residence. While our seminar facilities have hosted courses and course sections from many departments in the past, the Threshold project encourages computer–related courses in a wide variety of disciplines, particularly in those fields which have not until recently used the computer as a basic teaching tool. Examples include history, economics, education, statistics, sociology, and English. With our access to mainframe units in both the School of Arts and Sciences and in the Engineering School, students can use computer capability beyond the capacity of the PC. Almost any discipline, therefore, has the potential to become part of our curriculum.

Our facility serves 4 or 5 computer related courses each semester. These courses may be a small section of a large course, such as economics, or a smaller individual course such as "History and Computers" which was developed with the Hill Computer Room in mind. By prior arrangement with the Registrar, Hill students are given priority in registering for Hill courses. The advantages of such an arrangement are many: teachers can make their materials available in the Hill computer room for students to use outside classroom hours, students who live together can easily meet to work on class projects, and students and teachers can meet informally after hours or for meals in the Hill Dining Commons. This gives teachers and students an opportunity to work together and to get to know one another more easily than often is common in a big University such as Penn.

Not only does the Hill facility provide a place for students to work on Hill classes, but it also provides computers for a wide variety of other classes, including engineering, computer science, decision science and other social science courses. In fact, many students in the Humanities use the room only for word processing, and to encourage them to get started, we offer a number of two–hour seminars in word processing at the beginning of each semester. The facility, therefore, serves Hill students for a wide variety of purposes.

The Hill computer room is managed on a day–to–day basis by the Hill Operations Manager, a sophomore undergraduate student. This student supervises and trains his staff of about 10 undergraduates who serve both as consultants for the student users and security guards for the room. The Operations manager also keeps track of paper, ribbons and equipment in the room and reports his needs for supplies and repairs to me. Because this Manager is a student, he is aware of student needs and can often respond to their requests, for example, by extending hours for the room when he sees that homework projects for large classes are due on a certain day. Many of the other decisions about the computer room are made by me, for example, I recruit faculty to teach the courses, and I handle the budget, but I usually talk to the Operations Manager about everything. Since I report to the Faculty Master, the senior person responsible for the House, I discuss any major decisions about policy or equipment with him, as well as the student Operations Manager. Perhaps the bottom line of our philosophy of Hill House is that students are consulted about just about everything we do, and the Computer Room is no exception.

Many of our policy decisions are dependent upon the particular needs of the classes which meet in Hill. When we first opened our computer room last year, student coursework required that half of our PCs be hooked up to the Mainframe computers in the Engineering School and the School of Arts and Sciences. Now, partially because the Engineering School has redesigned their courses, we find that there is more of a need for students to use the PC's as isolated workstations rather than to use them as terminals for the larger computers, and we have modified our connections accordingly. Certain classes which use large databases, such as Sociology and History, will always depend upon access to the mainframes, but we have found the general trend has definitely shifted toward isolated PCs in the last two years. Of course in the future, we intend to modify our the number and kind of our connections whenever our needs change. Perhaps the optimum solution may be possible in the not-too-distant future when it becomes feasible for all computers on campus to be connected permanently.

Recruiting faculty to teach courses in the House is one of the most important parts of managing the computer facility. Because we are a freshman House, we look for teachers who give courses for the beginning student, and because our facility is fairly small, we are limited to a seminar or course section of about 20 students. We will accept courses in programming, but our primary interest is to find courses which use the computer as a tool in teaching a given subject. We have found that most teachers, if their course is appropriate, are willing to teach in the House. Because we value the good will of the faculty who teach here, we try to accommodate them when we can by scheduling the room to their wishes and by making books and discs available in our library for students to check out in the room. Faculty who are now developing course materials and software for many of these subjects, such as History or economics, find that we are a good place to test courses for the first time because our courses are small and teachers and students can easily talk about problems as they develop. Another advantage for faculty is that, because we limit access to students who live here or to those students in Hill courses, students almost always find a computer available when they need them.

B. INDIVIDUAL POLICY ISSUES

The Hill Computer Room is open from 10 a.m. until 1 a.m. every day except Friday and Saturday when we close at 10 p.m.; hours are extended when major projects are due in the more popular classes. During all hours when the room is open we have a staff person present who checks out discs and books, who makes sure that students are residents in the House, and who enforces rules for the room. The computers are housed in security devices and we have an alarm system on the room which we use during vacation periods. We have considered opening the room without the library of discs and books for after hours use, but we've been so lucky in avoiding repairs and vandalism, we hate to tamper with success. As I mentioned before, our schedule is modified to accommodate times when many students have projects due, and I haven't heard many complaints about the hours or availability of machines.

Unlike other computer facilities on campus, our computer room serves students from virtually every academic discipline at the freshman or sophomore level. Our policy about access has changed over time, but currently the room is open to any University person before 6 p.m. and after 6 the room is restricted to Hill residents and students enrolled in Hill Course sections. Because our usage is very heavy at night and is much less during the day, we require that courses and labs for Courses be scheduled before six. Our policy up until now for student access has been first–come, first–serve for Hill Residents, with the understanding that game players must relinquish their computers to people with serious academic work. Word processing is given equal priority with other kinds of work, but people using these programs are asked not to use either our two hard–disc machines or the computers which are connected to the mainframe.

Game playing has been the issue we've discussed most this year. Probably the simplest and ideal solution from the standpoint of the faculty and staff in the building would be to outlaw games altogether. Quite a number of students in the House, especially the Graduate Fellows on staff and the more serious students in the House, did voice a concern at the beginning of the year that games are disruptive and should not be permitted at all. We realize, though, that game–playing is the reason many people get started on computers in the first place and we do want to encourage students to use them. So, working with many of the game–players and the anti–game players, we came to a compromise: games would be permitted, but if noise prevented a student from working, the consultant could ask that game players leave. To keep the machines from contributing to the noise, we installed switches to turn off sound on the machines. Having the room staffed with a student consultant is an important factor in solving this particular problem since many students in a freshman house may not have enough courage themselves to ask the athletics types to turn down their game of computer football.

The area I would most like to improve next year is in training our staff of students who serve as consultants and security guards. We have tried to the extent that we can to hire students who are already knowledgeable in computers, but with the long hours that we must staff the room, we can't always be choosy in terms of hiring. Given uneven talents, training obviously becomes even more important. Ideally, I would like the computer staff to be able to use the printers, to use DOS commands, to connect with the mainframes, to read the directories on the hard discs, to use the basics of "PC–Write" and they should be able to start up some of the common languages, Pascal, Logo, APL and Basic. Since we now have enough experienced people in the House, we can train the undergraduates much more thoroughly next year.

Most of the costs involved in running the room are borne by Hill House. Students may check out books and discs and may use ribbons and paper without charge, and because we have a room monitor, we may prevent students from taking a large amount of paper for their own use. One of our most serious concerns, of course, is trying to protect copyright laws in accordance with University regulations. One one hand, we try to make available those programs which can be copied legally, such as "PC–Write," and, because, for example, this one is free and can be used on many computers around campus,"PC–Write" is the word–processing program we have encouraged. On the other hand, in an attempt to discourage students from pirating copyrighted material, our program library lends only those programs which the House owns or for which the University has a license. The problem is our two hard–disc machines, which constantly tempt the more seriously inclined computer students to share programs with each other by leaving them on the hard disc. Realizing that students are likely to do this regardless of signs and regulations, one of our more experienced students cleans off off extra programs and personal files from the hard disc each week. We therefore try as much as we can to limit students' to only the programs we own.

C. EVALUATION

In November of each year we conduct a House–wide evaluation Every student in the House receives a questionnaire asking him or her about the facilities, staff, and programs, including the computer room. While the faculty can easily see benefits of the students living and working together on their coursework, the students themselves most often mention more practical advantages such as not having to go out in the cold at night in order to get their computer projects done. The problems most often mentioned, noise and occasional crowded conditions have been alleviated by posting clearer rules, and getting the staff to enforce them consistently. Another concern, as I mentioned above, is that the staff has not been sufficiently trained enough to handle all the common problems which arise. In general, however, the room has been extremely successful in the minds of the students who live here.

Certainly the presence of such a facility affects curriculum. Because faculty can depend on students having a ready access to PCs in our building and because they can make materials available whenever our room is open, they can design their coursework to include computer projects on a regular and continuing basis. At least two courses, "Computers and Society" and "Computers and History" are most likely only being given because of our facility. Courses such as economics may cover the same general material as in the past, but with our facility, faculty can assign hands–on projects in simulating various economic conditions. The research students can be expected to perform for such courses as Statistics, Sociology and History changes totally when the students are equipped with the skills and equipment to deal with large amounts of data. Again, having books, discs, programs, computers, a classroom and the students all present in one building makes giving such classes much easier.

Our goal is to continue to consult with faculty from many different departments on an on–going basis, and we hope that we will be able to give some different courses every year. Because we consider our program in Hill to be an experiment, we constantly want to refine our policies to balance the needs of both students and faculty. We have considered our room to be extremely successful up until now and we hope we can maintain our good fortune in the future.

CONVIVIAL COMPUTING AT THE FORBES COLLEGE COMPUTING CENTER, PRINCETON UNIVERSITY

Presenter: M. Mahoney *Princeton University*

GOALS AND PURPOSES

Located in one of five residential colleges set up in 1982 to bring about a closer integration of the academic and social lives of the freshman and sophomore classes, the Forbes College Computing Center (FCCC) serves quite general purposes. It is designed to provide members of the college with access to the range of computing currently available at Princeton and gradually being introduced into its undergraduate curriculum. Through a program of formal instruction and a staff of student clinicians, members of Forbes can receive an introduction to the computer and to the possibilities of its use as a tool for learning. We hope that as the facility becomes an established part of the life of the College it will encourage students and faculty to explore together new applications of computers to education and will provide some of the technical support needed to develop and maintain those applications.

FACILITY AND GOVERNANCE

FCCC consists of 29 XT/370s (640K) and one AT linked via PC/VM Bond to an IBM 4321 minicomputer acting as a file and print server and providing passthru via phone lines to the University's Computing Center (PUCC) and the mainframe there. The local fileserver runs CMS on VM and provides local and system message and mail facilities. Other equipment includes a XXXX lineprinter, a PC laser printer, two color graphics printers, two graphics printers, and a plotter. Modems attached to two of the machines give students access to other computers on campus, in particular the VAXs at the Engineering School and at PUCC.

The XTs and associated peripherals are distributed among several several computing environments. A large classroom designed for lectures, demonstrations, and workshops contains twelve machines arranged close to one another in four banks of three and all facing in the same direction. A thirteenth machine for the instructor sits at the front, next to a large whiteboard. At present, the classroom lacks a video projector, which it probably should have for full effectiveness.

Two smaller workrooms hold six machines each. There the worktables are placed against the walls, and the computers face out toward the center of the room. In keeping with the goal of convivial computing each workroom provides an environment in which students can either work alone with no distractions in their field of view beyond the machine or collaborate with easy view of all the screens around the room. One of the workrooms is located next to the College's library, to which it should eventually become an adjunct by providing access to both online databases both within the University and outside it.

Forbes College had a tradition of student management of local computer facilities long before Project Pegasus began. During the '70s students at what was then Princeton Inn College set up, maintained, and administered a small cluster of terminals to the University mainframe. FCCC has continued that tradition. Student leadership consists of a Director, three Assistant Directors (for maintenance, software, and user services, respectively) who oversee the facility and in particular the work of some fifteen student clinicians, from whom the next generation of leadership will be drawn. A Faculty Director, charged with getting the project underway, has provided advice and guidance from the outset, but his direct role has decreased as FCCC has taken final shape. In the long run, the responsibilities of the position should be light enough to be assumed by an interested Faculty Fellow of the College.

FCCC has a budget provided by central funds to meet its requirements as part of the University's computer system. College funds supplement that budget, mostly for the purchase of hardware and software to support local options. (Just who would pay for phone links to external databases remains an open question at present.) Having students manage the budget has made them aware of the costs of the facility and has forced them to decide on their own priorities for computing. Although charged to the FCCC budget, maintenance and system support for the fileserver are the responsibility of PUCC, which consequently has oversight and veto over its uses.

INDIVIDUAL POLICY ISSUES

FCCC is open daily from 7 A.M. to 1 A.M. Those wishing to work overnight may sign out keys. If proper security can be established, the facility will eventually be left open around the clock. Clinicians are on duty in pairs during the evening hours. Their duties consist mainly of tutoring and trouble–shooting, although they are also responsible for enforcing the rules. At other open times, FCCC is left "unguarded", although its location in the basement of a building with students and service personnel generally about affords some protection, as do cables and alarms on each machine. So far, misuse has posed no problem, perhaps because the users all live in a small community and know one another.

In part, FCCC is an experiment to determine whether easy and reasonably assured access to computing will encourage students to use the computer as a general tool for learning. Density is perhaps the major factor in accessibility. The ratio at Forbes College is roughly one workstation for every twenty students, which should suffice to avoid long lines except at the times of highest contention toward the end of the semester. To maintain control over that experiment, the project has so far granted access to FCCC only to Forbes students and students in courses using the facility for specific purposes.

As a facility designed to support general use by an undergraduate community, FCCC's local system software consists largely of generic packages, including DOS, WordPerfect 4.1, Lotus 1–2–3, Turbo Pascal, BASIC, and a locally written LISP. These reside on the hard disks, which are regularly reformated from master files residing on the fileserver. Some of the machines have cards and software to support APL. Although PC/VM Bond makes it possible to run PC software from virtual disks on the 4321, experiments with PLATO have shown that most processes, in particular graphics and transfers of large files, run too slowly to make that an appealing option.

As a general, open facility, FCCC leaves it to students to decide how they want to use it. Word processing is fostered, supported, and encouraged. Games are permitted but not provided. Copyrights are protected by site licenses, dire threats, and regular reformatting and purging. The worst problem so far is pirated game software; students using it face expulsion from the Center.

EVALUATION AND FUTURE PLANS

Educationally, the success of Forbes College Computing Center has been at best mixed. On the one hand, the students are using the Center in steadily growing numbers. By late fall, a majority of the students —–including two thirds of the freshman—– had logged on to their fileserver accounts at least once, and we have reason to believe that many more have used the machines without using the fileserver. During midterm week and reading period, the Center was filled to capacity throughout the evening.

On the other hand, most students do not appear to be using the computers for anything other than word processing, nor have they shown much interest in exploring other uses. Moreover, to call what many of them do "word processing" is to lend a computational tone to an essentially clerical operation: they are using the machines as electronic typewriters. Relatively few people seem to be storing their papers on the fileserver, or indeed storing them at all for any longer than is necessary to get them typed and printed. Only recently has there has been any call for Lotus 1-2-3 or for SAVVY, although their availability has been advertised and classes have been offered.

One can hardly blame the students for their lack of interest. From the outset, we have held it as a principle that students will learn to use the computer whenever it is to their clear and present advantage to do so, that is, whenever it facilitates their work in evident ways. For that reason, we tried to design the educational program around academic tasks: writing a Lit 151 paper, writing a research paper with footnotes, and so on. So far, no persuasive tasks beyond writing papers have emerged. One can think of how students might use Lotus or SAVVY, but no one among actually uses the these packages for such purposes, nor have any courses in the University presented students with appropriate tasks. Powerful applications packages are intricate and often difficult to learn. Learning them has no independent virtue. Unless one has a clear need for their power, they are not worth the time and effort needed to apply them to even the simplest tasks. Hence, students are right in ignoring their availability.

Lack of a compelling curriculum is one reason why the educational program for this fall did not materialize as contemplated. Staffing is another. Had we wanted to be more aggressive in offering lectures and workshops, it is not clear where the instructional staff would have come from. Only a few of the clinicians had the technical facility --not to say the skills of presentation-- necessary to teach the various packages available at Forbes. The classes that did take place showed that, especially when the audience is uncertain of the value of what is being presented, demonstrations must be structured, clear, and well paced. Doing the job properly means devoting significant time and thought to both content and style. Before making that investment, both financially and personally, it would help to be sure of its long-range benefits. For the moment, at least, they are far from clear.

Moreover, contrary to initial expectations it does not appear that students will constitute an independent, innovative force for the introduction of computing into the curriculum. Nothing like that shows signs of taking place at Forbes, nor (to my knowledge) in any of Princeton's Pegasus projects. It is far more likely that the computer will move into the curriculum through the normal channels of development and innovation by faculty. Indeed, curricular applications will make most sense academically and intellectually where they reflect faculty use of the computer for their research.

DESIGNING LOCAL AREA NETWORKS FOR THE ACADEMIC ENVIRONMENT

Presenter: Stuart Bogom *University of Pennsylvania*

This presentation has two main themes: 1) describing the implementation of a LAN at the Population Studies Center of the University of Pennsylvania and 2) emphasizing the importance of management in the implementation of LANs.

The Population Studies Center is a demographic research institute affiliated with the University of Pennsylvania. Work at the Center focuses on the U.S. population (both current and historical), as well as populations in less developed countries, especially in Africa. Our staff includes 12 full time researchers, some 2 dozen affiliated researchers, 35 graduate students in residence, 2 full time computer support staff members, and 2 full time secretaries.

Mainframe computer use has been pervasive in demographic research for many years. While the manipulation and analysis of large census-scale data sets will continue to take place on mainframe computers for the foreseeable future, microcomputers have been rapidly accepted by demographers at the Center, and are used in a growing number of ways. In part this is because of the general usefulness of high power workstation in all academic work. In particular, microcomputers are very attractive tools for all-purpose demography in less developed countries where access to mainframes is often problematic. The Population Studies Center has been deeply involved in the transfer of demographic computer technology to Africa and elsewhere. In these environments microcomputers are often the only available tools for research. Therefore, our researchers and students have actively incorporated microcomputers into their work.

At the Center itself, microcomputers are used for three major functions: research, education, and office automation. Their needs vary. Research microcomputing requires mainframe access, file transfer between micro and mainframe, use of a wide range of microcomputer software, and analysis of data sets up to 10 megabytes in size.

Educational microcomputer use includes access to the mainframe, file transfer, use of a medium-sized range of software, and analysis of the same data sets used by faculty members. Office automation use consists primarily of word-processing support for researchers and secretarial staff.

A little over a year ago, I decided to introduce a LAN into the computer environment of the Population Studies Center. We had successfully experimented with a small, limited function system for sharing hard disk storage. I became convinced that, even though my users did not yet know it, they needed the increased function of a state of the art local area network.

My original basis for this decision was the flood of articles in the trade journals, almost all of which declared that LANs were the wave of the future in microcomputing. Their rosy prognostications (based, I am sure, on as much experience as I had myself) promised that LANs would connect everything to everything, that users and computer managers would immediately fall in love with their LANs, and that no one would ever question their purchases.

What these articles rarely mentioned, and what I only dimly perceived, was that LANs have a down–side. It is possible for a LAN to become an expensive waste of time, energy, and money causing more problems than it solves.

Somewhere in between the promises and the disasters lies the reality of most LAN installations: some problems are solved, and some new ones are created. As I installed and developed our LAN it became clear to me that given a technically adequate set of hardware and software tools, the quality of LAN management is the factor that makes the difference between the success or failure of an installation.

The first, and perhaps most important element in managing a LAN is assessing the users' needs. There is an added difficulty to this when your users are not asking for the LAN to begin with. Nevertheless, my own perception of their needs (in order of priority) was the following:

1) Most important in the short run was software delivery. My users typically employ a wide variety of software tools. Maybe two dozen are in steady use. The problem of tracking disks, distributing updates and managing availability had become significant and was growing rapidly. A network would help me as software manager, and my users as software consumers.

2) The ability of LANs to share expensive peripherals also seemed important. In order of priority, the devices of interest were letter quality printers, a plotter,

and perhaps a modem or two. Before the LAN, using these devices was awkward and inconvenient.

3) Mainframe terminal access is important for almost all of my users. Those with microcomputers in their offices have twisted pair wires installed to give them 9600 baud asynchronous communication to our mainframe. This solution is neither elegant nor cost–effective. With a LAN, I hoped to establish one physical connection to the mainframe which could be shared over the network.

4) File sharing is relevant to my environment in two ways. First, there is some need for joint access to research data files (e.g. a faculty member and a graduate student collaborating on a study). Secondly, there is a need for researchers and secretaries to both use a single copy of a document.

5) Last in priority, I saw a possible need for electronic mail. The Center offices are all on the same floor of a building, and communication is usually easiest by voice or phone. But I anticipated some need for another mechanism.

In addition to assessing these needs, having a sense of user familiarity with microcomputing is important. An environment appropriate to a novice is not necessarily adequate for a "power user." In my case, which is probably typical, I have a wide range of users covering the spectrum from the Third World student who has never used a computer before to professional programmer analysts.

When planning and installing the Center's LAN (an IBM PC Network), I made the following design decisions in order to adapt the LAN to my user's needs:

A fundamental decision was to give my users the choice of whether to use the network or not. I installed LAN adapter cards in every PC for which I had equipment, attached their cables, and supplied them with the necessary software and training. But I never told them they had to use the network. If they were going to use it, I wanted it to be because they found it useful.

This strategy had an additional benefit for me: I would be able to judge the effectiveness of the LAN by how many users chose to connect to it during the day and I could track its most successful functions by those programs used most. If few bothered to start the network, I would know that it was easier for them to work without the network.

Secondly, I decided to make the network as transparent to the users as possible. They had all used PCs before the network and were fairly comfortable with them. I made sure that nothing I did to their machines would interfere with the way they were accustomed to using them, in order to avoid a sense of something being taken away. The point was to provide added function.

In the area of server distribution, I took advantage of PC Network's capacity to have multiple network servers. In this way different levels of function were provided to different areas as appropriate. I use one AT as a primary software server to which all users have access. Office automation functions are a somewhat distinct activity at the Center. The primary users are the secretarial staff, with occasional access throughout the day by researchers. The services needed by the secretaries are high–quality word processing, document storage, and high– quality print services. To accommodate their specialized needs I installed a 20 megabyte hard disk in one of their PCs and configured it to provide word processing, document storage, and print serving just for the secretaries.

Security is an issue for office documents. I addressed this problem by installing a keylock on the power switch of the secretarial server. Each staff member has access to a subdirectory on the secretary's disk, and that access is password protected.

An additional server is configured for limited use in the storage of large data sets. In the near future I am planning to install a server for a software product which is very popular among my users and very resource intensive – SAS for the PC. I will also be setting up a server for external communications.

In considering the direct user interface of a DOS based system the choice is always between the "A>" prompt and a more friendly shell (locally written or purchased). I chose to go with the less friendly prompt, for the following reasons. First, my users are expected to develop a reasonable degree of expertise with computers, both micro & mainframe. There was no point in babying them, they would have to learn to master operating systems less friendly than DOS. Secondly, user shells consume significant amounts of memory, and even with fully packed PC's memory can become a constraint in a network environment when running large applications software.

The memory issue is a serious one – not so much with word processing applications, but with analytical ones. For instance, my users are starting to explore SAS for the PC, a popular mainframe statistical package that has been rewritten for the microcomputer environment. SAS requires 450K free memory. Most of my users run the IBM PC Network Program with the receiver configuration, requiring about 110K memory (including the resident portion of DOS). There is simply no room left for DOS shells such as Topview or Microsoft Windows until DOS's 640K memory barrier is broken in a definitive way.

The final significant design feature of the Center's PC Network is the approach to software access. All network users are provided a disk to use for starting the network. This disk contains the IBM PC Network Program and a batch file which:

1. starts the network
2. attaches a network disk
3. passes control to a batch file on that remote disk
which establishes other network links and tailors the DOS environment
4. displays an informational screen
5. provides an exit to a user written batch file to
allow automatic execution of user-specific commands.

Tailoring the DOS environment includes setting a path of subdirectories to be searched for command files. A design goal was to minimize this search path. It contains 3 components: a remote subdirectory of batch files which control access to software available on the network, a remote subdirectory containing DOS external commands, and a remote subdirectory with PC Network software.

This arrangement has several consequences. First, the user does not have to know how to manipulate the network in order to use the software on it. In fact, the user does not have to know anything about the network. When the command SC3 (to start SuperCalc 3) is entered at the keyboard the current drive/subdirectory is searched for a file with this name (this allows the user to run from a local disk if preferred). If SC3 is not found locally, DOS follows the search path and locates the file SC3.BAT in the network's batch file subdirectory. This file begins execution and invokes the application from wherever it is located on the network. Depending on the nature of the application, this process may include attaching and detaching specific disks to the users machines, but this process is transparent to the user.

This design keeps the search path short, therefore minimizing network activity necessary to locate software. It also discourages accidental interference with network software because the user does not necessarily know where it is located (as a general rule, all software disks are shared as read–only whenever the software allows).

While the design and implementation of the Center's network delivers needed services, and increase my users' computing freedom and flexibility, it is not without its difficulties. Here are two examples of managerial – not technical issues – which are problems in our network environment.

The first is the attachment of a laser printer to the LAN. We have one laser printer at the Center, and its primary use is by the secretaries. The secretaries share a large, centrally located office, and the laser printer is attached to one of the secretarial PC's that is configured as a server on the network.

Everyone would like to have a laser printer. They are quiet, fast, produce beautiful documents, and are quite expensive. The network hardware and software will allow any network user access to our single laser printer. However, consider the problems that would ensue.

A secretary, in preparation for printing a letter, has loaded Center letterhead paper in the printer. Before she returns to her desk, a researcher in an office down the hall starts printing a 15 page article – onto the letterhead. There are two difficulties with this scenario: the secretaries would become printer operators (a task in which they have no interest), and they would lose control over an essential aspect of their work environment – the letter quality printer.

For the introduction of a new technology succeed, it must bring real benefits to its users. It was clear to me that granting universal access to the laser printer, while helpful, was far outweighed by the disruption it would cause for the secretaries. The solution implemented, while not optimal, manages to increase everyone's flexibility somewhat. That solution is to give every researcher access to a personal subdirectory on the secretaries' server PC. This is the same subdirectory in which the secretaries store documents for that person. When a researcher wants a document printed on the laser printer, they transfer it to their subdirectory on the secretarial PC and send a message to the secretary requesting it be printed. While not the most elegant solution, it improves upon walking down the hall with a floppy disk.

A second example is the Center's plotter. Again, this is a very nice peripheral to have available, and, again, the network allows it to be shared with every user. However, the problem in this case is that plotter requires paper loaded by hand, and the correct color pens inserted. There were no candidates for plotter operator. The potential trouble and inconvenience caused by sharing the plotter on the network far outweighed the usefulness of universal connection to a lightly used device. Therefore, the ultimate decision was to put the plotter in a common room where users could bring their floppy disks and use it when needed, but not share it over the network in any way.

What problems and deficiencies exist in my network? There are no major outstanding problems, but there are some goals I have not yet achieved.

Top on the list is efficient communication with our mainframe. I am still using the same method in use before the network was installed: twisted pair wires connected to each PC for asynchronous communication. My goal is to use the single network cable for all communication. Developments in this area are being announced rapidly, and I hope to have a solution in the next six months.

Another problem (and frustration) is the presence of remote IPL capability on the PC Network adapter card, and the lack of any software to take advantage of it in a usable way. In an academic environment the "diskless booting" that such a feature allows would be a tremendous convenience.

The PC Network software needs improvements in the handling of messages – as currently implemented its functionality in this area is limited. Even better would be a true electronic mail facility.

The problem of software licensing in a network environment is one which must be addressed by software vendors. Only a few of them have risen to the challenge. It is important that the vendors confront this issue as soon as possible

Finally, in the Center's environment, I am beginning to address the issue of performance. Tools are available from IBM and other vendors to tune the PC Network. Thoughtful design of the network environment also helps improve performance. But as the applications grow larger, and their need for disk I/O increase, the need for improved network performance will increase. While it is not yet a problem, I anticipate that within a year's time I will need a high speed, high capacity server on my network of a type that does not yet exist in the marketplace, to the best of my knowledge.

On the whole, the introduction of a local area network to the Population Studies Center has been well received. Over the last six months use has grown, and people not on the network are now asking me what they can do to get connected. As new software & hardware products emerge to take advantage of LANs, the usefulness of it will increase.

I am also certain that if mismanaged, the introduction of the Center's LAN would have been a disaster. Assessing user needs carefully, and making appropriate design decisions in a thoughtful way such as I have described made the difference in my installation. I hope that these experiences help you to have the same success.

NETWORKING TO SUPPORT EDUCATION OBJECTIVES

Presenter: Michael K. Stenstrom *University of California, Los Angeles*

INTRODUCTION

In 1984 the UCLA School of Engineering and Applied Science (SEAS) began to develop a new schoolwide network (SEASnet) to provide computing and communication services for its six departments (Chemical, Civil, Mechanical–Aerospace–Nuclear, Materials Science and Computer Science). We envisioned a network which allows users to perform their research and teaching as well as communicate, all through computers.

We had two existing models to build from. The Office of Academic Computing (OAC), which had previously been the exclusive supplier of computer resources to five of the six engineering departments (exclusive of Computer Science), operated an IBM 3033 running MVS supporting TSO and Wylbur. Each department had a limited number of 3278 style terminals which provided access to higher–level departmental users. Undergraduates accessed the 3033 through centrally located terminals at OAC. Computing was restricted due to the overload on the 3033, as well as the need to recover operating costs through recharges.

The other model was a network of 15 to 20 DEC VAXes in the Computer Science Department. Some of these machines were running Berkeley Unix (BSD 4.1) while others were running a new operating system, Locus, being developed by members of the Computer Science Department under NSF and DARPA funding. Locus was developed from BSD 4.1 with the objective of providing distributed computing among network of heterogeneous machines. Among its objectives were a distributed file system with one namespace, distributed computing among any number of machines, and machines dedicated to specific tasks (servers). Transparent access to one's files and computing resources with automatic site and server selection was envisioned.

The timely arrival of Project Advance provided funds to develop the school's network. The second model, distributed Unix, was selected. Strong emphasis was placed upon the workstation/server concept. Long term goals were to provide free computing to academic users without burdening them with awkward computing concepts, such as MVS JCL. Locus with Unix and DOS workstations was selected.

EARLY PLANS

A committee representing all six departments was appointed to develop the new network. Early deliberations centered around the need for Unix, workstations and terminals, and supercomputing––all conflicting objectives to some extent. The Computer Science Department was heavily invested in terminals and time–sharing minicomputers. The non–Computer Science Departments knew very little about Unix. No Unix or advanced workstations were available from IBM Project Advance.

In this uncertain environment, a plan was created which would utilize the advanced features of Locus to merge IBM hardware and DEC hardware into a single network which would look and function to the naive user as a single large Unix computer. Project Advance funds were budgeted for the yet–to–be–announced IBM workstations.

TCP/IP Ethernet was selected as a networking protocol. This provided a mechanism for heterogeneous hardware to communicate using a non–proprietary networking protocol. The same hardware and cables could convey the Locus host–to–host networking traffic, as well as the workstation to server communication.

PLAN MATURATION

As software developed and new hardware was announced, plans matured into the network shown in Figure 1. Two separate Ethernets were constructed. The first was a host–to–host network, which provides communication among Locus servers, and communication from TCP/IP workstations to servers. The second provided terminal connections through Bridge Communications servers. In this fashion, the heavy investment in terminals and time–sharing computers was preserved. (See Figure 1. SEASnet)

An IBM 4381 and two 4361's were acquired, and complement the existing VAXes. An FPS–164 was obtained to provide advanced floating point resources. Two BSD 4.2 machines were acquired to support special net interest. Gould donated a PN–9080 to support VLSI and Graphics research, and the

Electrical Engineering Department purchased a Pyramid to support its own special needs.

The PC–AT was selected as the low level workstation. Two classrooms of 30 PC–ATs each were constructed. The classrooms were designed to serve two purposes: laboratory and recitation in the traditional fashion, and the new concepts of computer aided instruction where computers are integrated into lecture classes. A number of machines were located in faculty offices, and a number of small departmental "clusters" were created.

Personal Computer Interface (PCI) was acquired to provide transparent access from PC–DOS machines to Locus servers. PCI allows each PC user to access files on any of the server machines through conventional DOS utilities and to submit jobs from workstations to the server machines. In this fashion even the non–Unix users can access the net and its server machines.

Terminals access the 4300's and the array processor resources through the VAXes. Locus with its distributed processing allows a user logged into one of the VAXes to execute processes on one of the 43XX, or any other Locus machine. The only requirement is permission on the target machine. Files can be and are often remotely located. The 3380's are an economical way of providing large quantities of disk storage to VAX users. The VAXes handle the character input/output and free the 43XX's from costly, cpu consumptive interrupts.

At the time of this writing we are obtaining our first RTs. They will be added to the network and will be our first general purpose Unix workstations. A number of Unix workstations (Apollo's, Sun's and Microvaxen) exist within SEAS, but are dedicated to special user groups or research projects. We envision using mostly BSD 4.2 RT's because of the availability of TCP/IP networking protocol. A third classroom or laboratory of 30 RT's is being constructed, with an expected completion date of December, 1986.

Also at the time of this writing we are extending our network to OAC's new mainframe, an IBM 3090/200. This new machine was acquired on the basis that one of its tasks would be a server machine to SEASnet. It is the only general purpose supplier of MVS and VM/CMS software to SEASnet. We

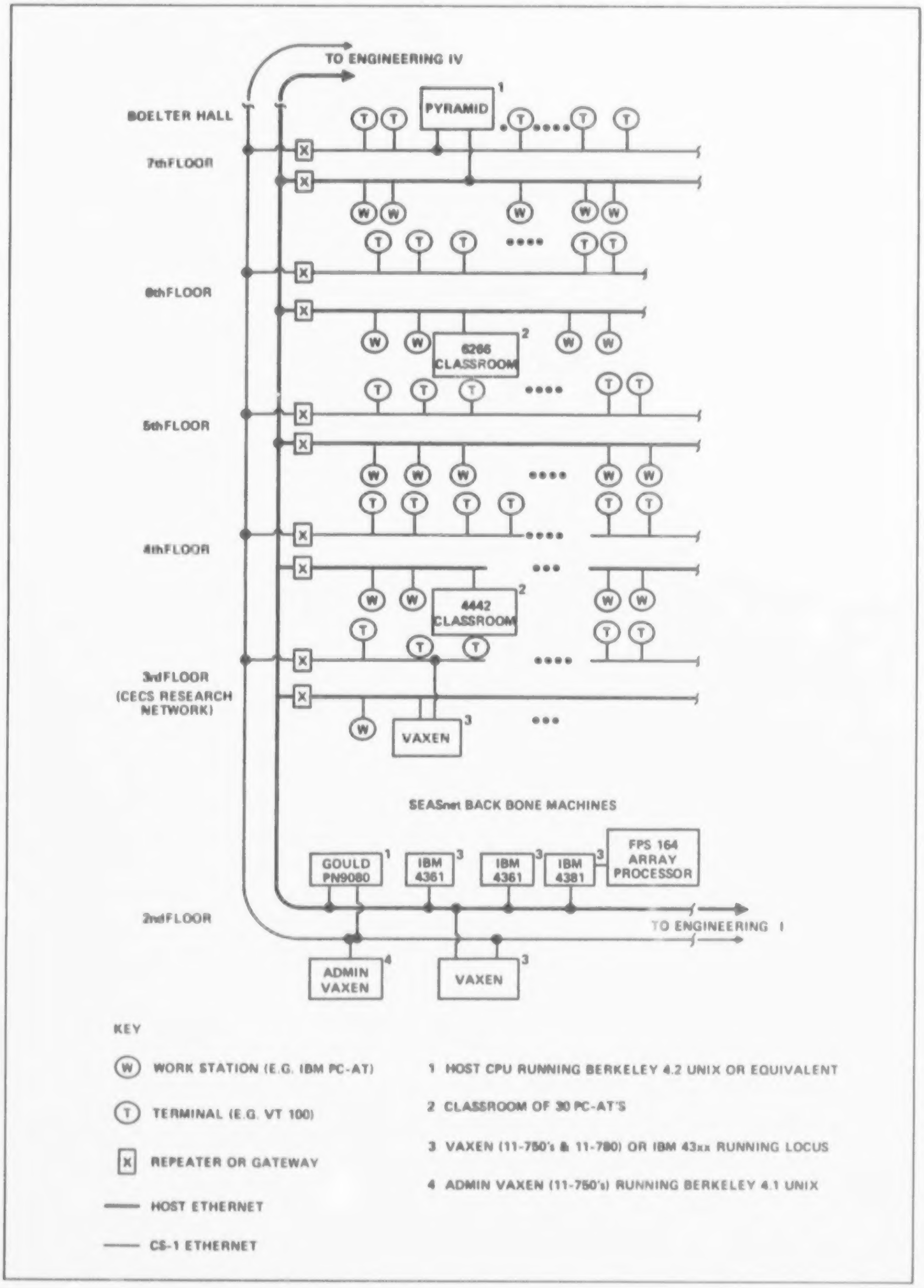


Figure 1. SEASnet

envision a continuing need for MVS and CMS software to selected SEASnet users. The 3090 also provides access to the University Library card catalogue.

The connection to the 3090 is being made in two ways. Terminals and PC's emulating terminals will obtain access through 7171's. An Ethernet to 7171 connection is made through Bridge Communication Servers. Thirty–two ports will be allocated to the host–to–host network, with 32 more existing on the CS–1 network. A TCP/IP interface is being developed in conjunction with OAC staff to handle file transfer and mail.

Additionally, a 4341 used exclusively to support the Manufacturing Engineering Program is being networked to SEASnet through an IBM channel to DEC Unibus/Ethernet adapter. This will allow the AT classrooms to access the large collection of CAD/CAM software available under VM/CMS.

SUCCESSFUL SOFTWARE AND COURSEWARE DEVELOPMENT

By far the most important ingredient for the success of our network has been the existence of a large software base. Unix provides a large collection of bundled software. The PC–DOS machines have now acquired a vast collection of useful programs.

We currently support a number of PC–DOS programs. The IBM Professional Fortran is the most popular software product we have licensed. Turbo–Pascal, IBM C, MS Word, IBM APL, and IBM BASIC are also useful and popular. Kermit has become a standard program for dial–in users, and is supported on all the SEASnet machines.

A number of high level application programs have been purchased to support selected classes. Control theory programs, digital simulation, mechanical design and mathematical software have been extremely helpful.

Integration of computers into existing classes continues to be an onerous task. To facilitate this integration a relationship of faculty and Teaching Assistants (TAs) was created and is shown in Figure 2. Fortunately a significant amount of University funding was provided to hire additional TAs help with this task. (See Figure 2. Software Development Model)

SOFTWARE DEVELOPMENT GROUP

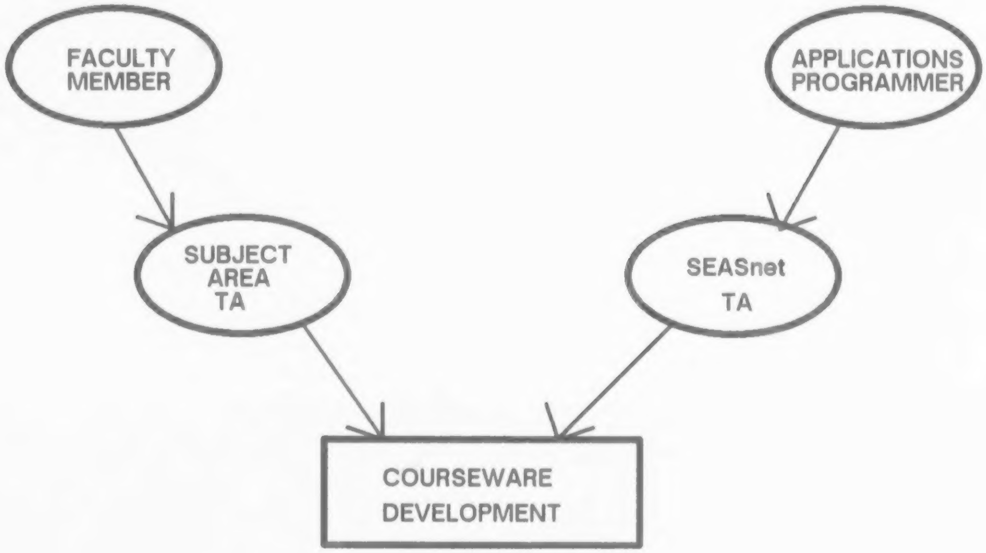


Figure 2. Software Development Model

In many areas there was no existing material and a number of programs have been written by faculty. Turbo–Pascal seems to be the easiest way to develop graphic interactive software.

Currently over 20 different classes each quarter are using the two SEASnet classrooms. 511 students used the classrooms in the Winter Quarter, 1986.

A number of tools have been or are in the process of development. Mail to PC's through PCI is under development. Menu generation programs have been developed to allow faculty to quickly and consistently write demonstration programs. High level graphics tools are necessary in order to allow students to incorporate graphics into their FORTRAN, PASCAL, or C programs. We have developed several such high level tools using GKS. The tools have had limited success, in part due to the overhead they cause when they are compiled or linked.

CONCLUSIONS

The use of workstations linked through transparent networking to server machines has revolutionized teaching in SEAS. More educational computing has occurred this year than in the past five years using the MVS TSO resources of a 3033. Students are more receptive to using distributed PC–DOS machines than centrally located timesharing services. PC–DOS is quite suitable for the lower level users. The inclusion of a Unix–like directory structure is its most important feature.

Software and Hardware issues are a continuing concern. The IBM channel to Ethernet interface is a difficult problem. All the methods available now have some severe limitations. Network licenses are generally not available, or too expensive. Appropriate software costs to the students must be some fraction of what they currently pay for books.

The development of this network provides a uniform path for entering Freshman to learn computing using PC–DOS machines, with which they are often familiar. As they progress, they learn Unix which allows them access to all the computing resources they need, from word processing such as troff, to supercomputing. Access to other operating systems is provided to those who need it.

PDS – A GENERAL AUTHORING SYSTEM FOR THE DEVELOPMENT OF SIMULATIONS IN POLITICAL SCIENCE

Presenter: **Mark Winer** *University of Maryland*

Coauthors: **Mark Winer** *University of Maryland*
Joe Oppenheimer *University of Maryland*

Research in public choice has shown that mathematical models, based on game theoretic concepts, can have broad applications to political science. Games are social interactions in which a number of individuals each make a choice. Together, the choices determine a collective outcome. In turn, the outcome yields to each one of the individuals some sort of cost, payoff, or reward. Often the choices involve strategic thinking, communication, coalitions, and the like. The theory has many real social analogues including elections, roll call votes, and committee decision making.

Recently, the authors and other researchers have conducted experiments that test the empirical applicability, predictive ability, and robustness of game theoretic models.

THE NEED FOR PDS

University faculty members frequently encounter difficulties finding opportunities to teach about the applicability of public choice politics. As researchers we often have difficulties finding the resources to design and develop our own empirical and experimental research. Political science faculty rarely have an opportunity to expose undergraduate political science students to mathematical models. Because of the student's limited mathematical background and the complex nature of the models, even the basic concepts and principals behind the models are difficult to incorporate into undergraduate courses. There is not adequate time to provide the needed background information. The only opportunity for an undergraduate to encounter even the basic principals of mathematical models and their implications is in special, upper level elective courses devoted to mathematical modeling.

Furthermore, graduate political science courses offer limited means to illustrate to students the applicability of these models or to permit students to develop, test, and analyze models of their own. As a result, many students never develop an intuitive understanding of the principals underlying the models.

Finally, experimental research is hampered because each new experiment requires the development of a new protocol, experimental design, and simulation tool. Often the process of setting up the simulation is so costly and time consuming that the experiments are never conducted.

As a possible solution to these problems, we are developing a set of software modules that provide the opportunity for a wide range of simulations of political activities and other collective choices. To serve many purposes, the modules must be flexible and user friendly. We plan to develop three modules: one simulating the production of public goods and general prisoner dilemma style situations, one simulating elections with a variety of ways to aggregate individual preferences, and one simulating committee decision processes under a variety of voting rules and agenda structures.

This paper reports on the first of these modules, PDS, a general authoring system that can be used to write simulations or models of many social or collective decision processes. PDS is a program designed to enable undergraduates to grapple with and apply mathematical models in a simulated political environment. This encourages students to understand the concepts and logic behind the models without having to understand their mathematical complexity. PDS can also be used by graduate and undergraduate students to develop and test hypotheses about collective choice. Finally, researchers can tailor PDS simulations to experimentally test a variety of collective decision making models.

THE PRINCIPALS BEHIND PDS

The PDS module is based primarily on prisoner dilemma games. The classic prisoner dilemma situation involves a group of individuals each faced with making a decision. If each acts in his or her own best interest the resulting outcome is definitely inferior.. That is, there is another outcome more advantageous to all participants. One major social analogue of the prisoner dilemma is a situation in which an individual must choose whether or not to incur private costs to support the procurement of shared or public goods. Public goods are different than ordinary private ones. Private goods go only to the person who buys them. When an individual purchases public goods they are supplied to the entire group. Two examples may help to illustrate the important role the provision of public goods plays in politics.

EXAMPLE 1: THE ARMS RACE SIMULATION

The Arms Race simulation is the simplest form of a prisoners dilemma situation. It involves two countries. Each must decide whether to escalate or curtail its production of nuclear weapons. Both countries, sworn rivals, would like to have a military advantage over the other. If country A escalates production and country B does not, then country A will be considered to be in a superior position. However, both countries would prefer not escalating to having both escalate – producing weapons is expensive. This situation can be illustrated by assigning numerical values to each of the possible outcomes, as in the table below. The higher the numerical value assigned to the country, the more that country prefers that outcome. The first number represents the numerical value to Country A, the second numerical value to country B.

	COUNTRY A	
COUNTRY B	escalate	don't escalate
escalate	10, 10	0, 200
don't escalate	200, 0	100, 100

The likely result of a situation like this is escalation by both countries. No matter what decision country B makes, country A would be better off escalating. Similarly, no matter what decision country A makes, country B prefers to escalate. Thus the dilemma, both countries would prefer the outcome "don't escalate" but both will choose to escalate.

EXAMPLE 2: AIR POLLUTION SIMULATION

A city currently suffers from extremely high levels of air pollution. Ten manufacturing plants in the area are each dumping large amounts of pollutants into the air. The Environmental Protection Agency monitors the level of pollution and finds that each plant is discharging 100 pounds of particulate pollution into the air each day. This is because all of the plants use soft coal as a fuel. The plants' profits for the year range from $2,000,000 to $5,000,000.

If the air quality is not improved to meet federal standards by the end of the year, the EPA will insist that each plant install special equipment that removes air born discharges to bring the pollution down to 600 pounds per day. The cost of the equipment depends upon its capacity: the cost is $100,00, for each 10 pounds of particulates removed. Thus, to completely remove all particulates would cost each plant $1 million. .The equipment each plant will be forced to buy will be enough to remove 1/10 of the excess particulates from the air as measured at the end of the year.

An alternative mechanism for removing particulates from the air exists: the procedure can wash the coal they are using as a fuel. The cost of this procedure is one half the cost of the special equipment. EPA does not use this mechanism because it is hard to enforce and monitor, but allows plants to voluntarily wash their coal. If the plants can voluntarily reduce the level of pollution below 60 pounds per day then EPA will not require special equipment.

Because many firms produce the same products as these firms, the costs of any cleanup would come directly out of profits. It can not be passed on to consumers as price increases. Thus, each plant is faced with the following situation: It would obviously be advantageous to wash the coal rather than install more expensive machinery. However, the cleanup costs for each plant depend on the total level of pollutants being discharged from all the plants. For every 100 pound drop in particulates, each plant saves $100,000. For each $50,000 a plant spends washing coal, it and each of the other plants saves $10,000 – a total savings of $100,000. Obviously it would be worthwhile for each plant owner to have the others clean up their coal sufficiently to have the total level of pollution fall below the EPA maximum. Unfortunately, all the owners have the same interest. Thus, the situation is somewhat similar to the Arms Race example. Each plant would prefer that all plants wash some of their coal. But, each plant also prefers that other plants wash a sufficient amount of their coal so that it can avoid any kind of pollution abatement procedures.

THE FLEXIBILITY OF PDS

Both of these examples and many others can be simulated using PDS. The flexibility of PDS allows the simulation designer to vary many of the parameters that govern the way individuals make decisions in these situations. The effect of these changes can then be analyzed. For each simulation the following characteristics can be varied:

1. The number of participants can vary from two to ten. One of the advantages of this software is that it enables the designer to construct prisoner dilemma situations and other similar situations involving the provision of public goods with more than the classic two players.

2. The number of rounds can be varied from one to ten. The designer can decide whether none, some, or all of the participants know the total number of rounds. In each round, many of the simulation characteristics described below can be varied.

3. The resource endowments of each player can take any numerical value and can be in any unit of measurement – dollars, cows, or nuclear weapons.

4. The participants can be asked to make two types of decisions. In one, the participants can be given a dichotomous choice: contribute to the public good/cooperative action or do not contribute. In the other, each participant decides how much of his or her total resources to contribute to the public good/cooperative effort.

5. The form of the production function for the public good can be altered by the simulation designer. The production function determines how the donations of the participants translates into a level of public good provided. If the simulation designer wishes it can be extremely complex. It can be a nonlinear function of the amount of resources donated by each player, with a variety of interaction constraints limiting or exaggerating the effect of some participants contributions. For most cases however, a simple linear produciton function is appropriate and this is the easiest for PDS to accommodate.

6. The designer defines the form of the utility or payoff function of each simulated participant. This function determines the return to each participant for each round. It is a function of the amount of resources the participants keeps, the amount the participant donates to the cooperative production of the public good, and the total amount of the public good produced.

7. Each player has the ability to communicate with other participants through a network of computers. With suitable hardware, PDS allows participants to send messages to all or some of the other players. The simulation designer can disable this message sending ability for any player so that the ability to bargain and form coalitions can be analyzed. In the Arms Race example, the participants' ability to communicate with one another can make the difference between both deciding to escalate or a cooperative agreement not to escalate. Communications require an Ethernet LAN and Ethershare software.

8. The amount of information revealed to the participants in the course of the simulation can vary. The simulation can be designed so that players know everything about the other players, the length of the simulation, and the outcome of all previous rounds. If the designer chooses, any part of this information can be suppressed. Thus, for example, in the Air Pollution example, each plant manager may know only his own profits but not the profits of the other plants. Or he can know the profits of all plants.

In addition to the wide range of parameters that can be varied, PDS permits the simulations to contain simulated players. The simulated players have preprogrammed strategies selected by the simulation designer. The number of simulated players can vary from none to one less than the total number of players. All simulations must contain at least one real participant. The simulated strategies available include: always cooperate, never cooperate, cooperate with a fixed percentage, and cooperate with a selected probability. More complicated simulated strategies are also available, including several versions of "Tit–for–Tat" (doing the same thing as some combination of other players did in the previous round) and any sequence of decisions entered by the designer.

The simulation designer can either use one of the prespecified simulations or can design his or her own simulation by appropriately setting each of the above options. Because the simulation design is menu driven, it is not difficult to author new simulations. Considerable effort went into such details as the self explanatory nature of the error messages. Undergraduate students were able to quickly learn the design process and develop their own simulations. Once the simulation has been designed it can be saved on disk and recalled for further editing. Thus, it is easy to develop slight variations in the parameters of a simulation and test for the effects of these deviations on the bargaining behavior and outcomes.

USING PDS

The design of the simulation determines the texts of the screens seen by the participant in the simulation. PDS is designed so that it is easy for participants to learn how to use the computer and take part in a simulation. There is always a prompt line at the bottom of the screen that tells the participant his or her range of options. In addition, detailed instructions for the simulation can be recalled and reviewed anytime. There are four types of screens that each player can access. Up to three different versions of these screens can be used in any one simulation. This is particularly useful when different participants are playing different roles. The simulation designer designs the screens and determines which are appropriate for which players.

The decision screen is the main screen for participating in the simulation. It describes the basic structure of the simulation including the type of decision to be made and the possible effect of different decisions. It is also the screen on which decisions are entered. From it, the participant accesses all of the other screens.

The design of the screen is completely free form, although there is a generic, default screen available. A number of variable fields can be located anywhere on the screen by typing the variable name in the desired location. The variable fields include the number of rounds, and each player's budget. Other screens are accessed by using the function keys: F1 for the worksheet, F2 for the payoff screen, F3 for the instruction screens, and F4 for the message screen. Hitting the function key again returns the participant to the decision screen.

A worksheet provides information about all of the participants. It identifies the participants by assigning each a letter and, if the designer chooses, also gives information about the budgets of all players. The worksheet can be used to review the effects of decisions made in previous rounds. The worksheet also allows a participant to enter hypothetical decisions for all players and see the effect of these decisions on his payoff and the payoffs of the other players. This gives players the opportunity to quickly see the effects of particular decisions.

The payoff screen provides more detailed information about the calculation of the payoffs. If players are confused by the abbreviated explanation on the decision screen, they can access the payoff screen. Like the decision screen, the player screen is completely free form, with a set of variable fields available representing characteristics of the simulation that may change from round to round or simulation to simulation.

The instructions for playing the simulation can be accessed at any time by hitting the F3 function key. The instructions may contain a description of the simulation as well as details on how to enter decisions and access and use the other screens. The first screen of instructions is the first thing that the participant sees. These instructions are created using an ordinary text editor and there is no limit on their length. The participant moves through the instructions by using the PgDn and PgUp keys. The player leaves the instructions by hitting F3 again. The ability of players to recall the instructions at any point in the simulation is one of the major advantages of PDS. Any player, confused about particular details of the simulation or having trouble using the computer, can recall the instruction screens and review the simulation description.

The message screen, accessed by the F4 function key, is used to compose and send messages to the other players. If the designer chooses, each participant can send messages to some or all of the other players. (Players are addressed by their letters, as seen on the worksheet.) The message can be up to 10 lines long. The previous messages a player received and sent can be retrieved, edited, and retransmitted. At any time, a player can see if there are any messages waiting for him or her by hitting the function key F5, or any other key. If there are messages, they will immediately appear on the screen.

The results of the simulation are automatically saved on disk. Detailed information about the way the players communicated and used the worksheets is also retained for the simulation designer.

CONCLUSION

As a teaching tool, PDS involves students in constructing, running, and participating in actual social experiments, It permits the cheap and easy construction of a professional quality lab for introducing students to scientific techniques in social science programs. Through experience with PDS, students can learn about the power of the mathematical models in political science. They can learn the basic principals underlying the models and develop an intuitive understanding of some of the important factors governing collective decision processes.

As a tool for teaching research methods, it permits even the beginning, or freshman student, to experience advanced social science research opportunities of generating and testing hypotheses. The students can easily generate professional quality social science data sets. The data is generated and stored in a manner which facilitates the application of statistical techniques. The research design underlying the data can be tailored to the field of interest of the course or the student. As such, it is a simple tool to increase the learning and creativity of the student.

As a research tool, PDS is a powerful general authoring system that can be used to write simulations or models of many social or collective decision processes. It facilitates the understanding of political and social events and behavior. The simulations can be tailored to test different theoretical and practical hypotheses.

Thus, PDS is a useful tool to political scientists. It aids in the teaching of the principals and concepts underlying mathematical models, in teaching research methods, and in designing and conducting experimental research.

REPORT ON THE BROWN / C–MU / MIT WINDOW MANAGER / ENVIRONMENT CONVERGENCE WORKSHOP

Presenter: **Andries van Dam** *Brown University*

Coauthors: **Andries van Dam** *Brown University*
Michael J. Braca *Brown University*

INTRODUCTION

One of the primary goals of the academic workstation software researchers at the Institute for Research in Information and Scholarship (IRIS) at Brown University, the Information Technology Center (ITC) at Carnegie–Mellon University, and Project Athena at the Massachusetts Institute of Technology is the sharing of scholarly tools developed at each site. At present such sharing is made difficult because each of the three sites uses a different software environment:

- ANDREW at C–MU/ITC
- X at MIT/Athena
- CADMAC (a "C" language Macintosh toolbox developed by Cadmus Computer Corp.) at Brown/IRIS

In an attempt to remedy the situation, Brown University hosted a Convergence Workshop, led by Professor Andries van Dam, in Providence, RI, on March 31st through April 2nd, 1986. The main goals of the meeting were to assess the technical and logistical obstacles facing convergence on a single software environment and to map strategies for overcoming those obstacles. Eighteen university representatives were in attendance, as well as two participants from IBM and one from DEC (the person from DEC being Jim Gettys who holds an appointment at MIT/Athena and is one of the co–developers of X).

CHARTER

All aspects of the software environment were subject to discussion, with the following constraints:

– Minimize religious issues. Note fundamental disagreements and move on to other topics to avoid becoming bogged down.

– Keep in touch with reality. Avoid prolonged discussions of "pie–in–the–sky" solutions and those that involve software with heavily constrained availability. It was noted that the software should not be restricted to any particular manufacturer's machines.

A REFERENCE MODEL

One of the first items agreed upon at the session was the structure of a window manager / environment reference model (see Figure 1), whose purpose was to provide a framework for discussion of proposed systems, as well as for synchronization of terminology. In the model, the user is shown connected to each application program (AP0 – APn) and to things called "Window Managers" (WM1 – WMn) which provide control of layout and sizing of windows on the display. The union of the interactions a user can have with the application programs and window managers in the system comprise the User Interface (UI).

Below each application program is an Application Programming Interface (API), of which we have identified 3 levels. At the lowest level (API0), the program will directly invoke the services of the "Window Management System" (WMS), which has three main functions:

- Provide a low–level hardware independent interface to the display and input devices
- Multiplex output to the display (via windows)
- Demultiplex input actions (event handling)

API level 1 is an interface to a graphics and window management toolkit which provides a higher level of access to the facilities made available by the WMS. For example, a set of routines for creating and controlling pop–up menus on the screen might be included in such a toolkit. More than one toolkit may be supported. Most of today's window–based applications run on this "intermediate" level.

Our feeling, however, was that most applications should be written to an even higher level, here labeled API 2, which is shown as an interface to an "Object

Oriented Programming Environment" (OOPE). Both Brown and C–MU are working with such systems; MacApp (from Apple) at Brown, and MOE (aka Base Editor II) at C–MU.

API 2 CONVERGENCE

The most important result of discussions of convergence at the API2 level was that everybody agreed that the object–oriented programming paradigm is the right path to better programmer productivity. An OOPE was seen to provide a much higher level API than conventional programming techniques, and to provide more general data sharing than is possible with the normal misinterpreted byte string "cut and paste" methods in wide use today.

A feature–by–feature comparison of C–MU's MOE system against the MacApp package used at Brown showed the two to be essentially isomorphic. This was especially good news since it shows that convergence is entirely possible at this level of API.

The goal was therefore established to work toward a single environment with a new name.

API 1 CONVERGENCE

It was also determined that convergence must occur at the lower level API as well to support a higher level merger. Fortunately C–MU has agreed in principle to switch to an Andrew emulation toolkit running on X instead of the straight Andrew wm itself (when performance and font support match current Andrew levels).

The basic approach agreed upon is to attempt to use the "best" features of each system, for example, use:

- Quickdraw for drawing baseline
- X for window management baseline
- union of Mac toolbox, Andrew and X facilities for toolkit baseline

The three groups will study the problems resulting from such a merger to try to determine whether fundamental changes need to be made to X to support it. Some problems may arise in adding the low–level Macintosh graphics primitives to the X WMS, although these are not expected to be insurmountable (masking of drawing operations with arbitrary "regions" or bitmaps leaps to mind). It was noted that we probably only have about 3 months to get such changes easily incorporated, since by then X will have been spread too widely. Speed in this area is therefore of essence.

CONCLUSIONS

Overall, the workshop was a great success in creating feelings of optimism in the participants that convergence is indeed feasible; especially exciting was the remarkable amount of agreement on what the higher-level API should comprise. Certainly the benefits of being able to share educational software tools outweighs the cost of converging to a common software base.

PARTICIPANTS

Mike Braca	Brown	Bruce Lucas	C-MU
Brian Chapin	Brown	Norm Meyrowitz	Brown
Tom Doeppner	Brown	Jim Morris	C-MU
Charlie Evett	Brown	Alex Morrow	IBM
Matt Evett	Brown	Ron Newman	MIT
Dan Geer	MIT	Andy Palay	C-MU
Jim Gettys	DEC/MIT	Tom Peters	C-MU
Fred Hansen	C-MU	Jerry Saltzer	MIT
Mike Kazar	C-MU	Bruce Sherwood	C-MU
Steve Lerman	MIT	Tom Stambaugh	Brown
Larry Loucks	IBM		

BIBLIOGRAPHY

A.Appel, W.Hansen, M.Kazar, B.Lucas, A.Palay: "The Multi-Object Environment Specification", C-MU/ITC, 1986

Apple Computer, Inside Macintosh, Addison Wesley, 1985

S.Bond, M.Critchfield, J.Kaminski: Andrew User's Guide, C-MU/ITC, 1986

S.Chernicoff: Macintosh Revealed, Hayden, 1985

F.Hopgood, D.Duce, E.Fielding, K.Robinson, A.Williams, editors: Methodology of Window Management, Springer Verlag, 1986

J.Gettys: "Problems Implementing Window Systems in UNIX", Proceedings 1986 Winter USENIX Technical Conference

J.Gettys, R.Newman, A.Della Fera: "Xlib – C language X Interface, Protocol Version 10", MIT Project Athena, 1986

J.Morris, M.Satyanarayanan, M.Conner, J.Howard, D.Rosenthal, F.Smith: "Andrew: A Distributed Personal Computing Environment", Carnegie–Mellon University ITC, 1985

D.Rosenthal, "A Programmers Guide to [Andrew] wm", C–MU/ITC, 1985

D.Rosenthal: "Window System Implementation, 1986 Denver USENIX Course Notes", USENIX, 1986

L.Tesler: "An Introduction to MacApp 0.1", Apple Computer, 1985

Window Manager / Environment Reference Model

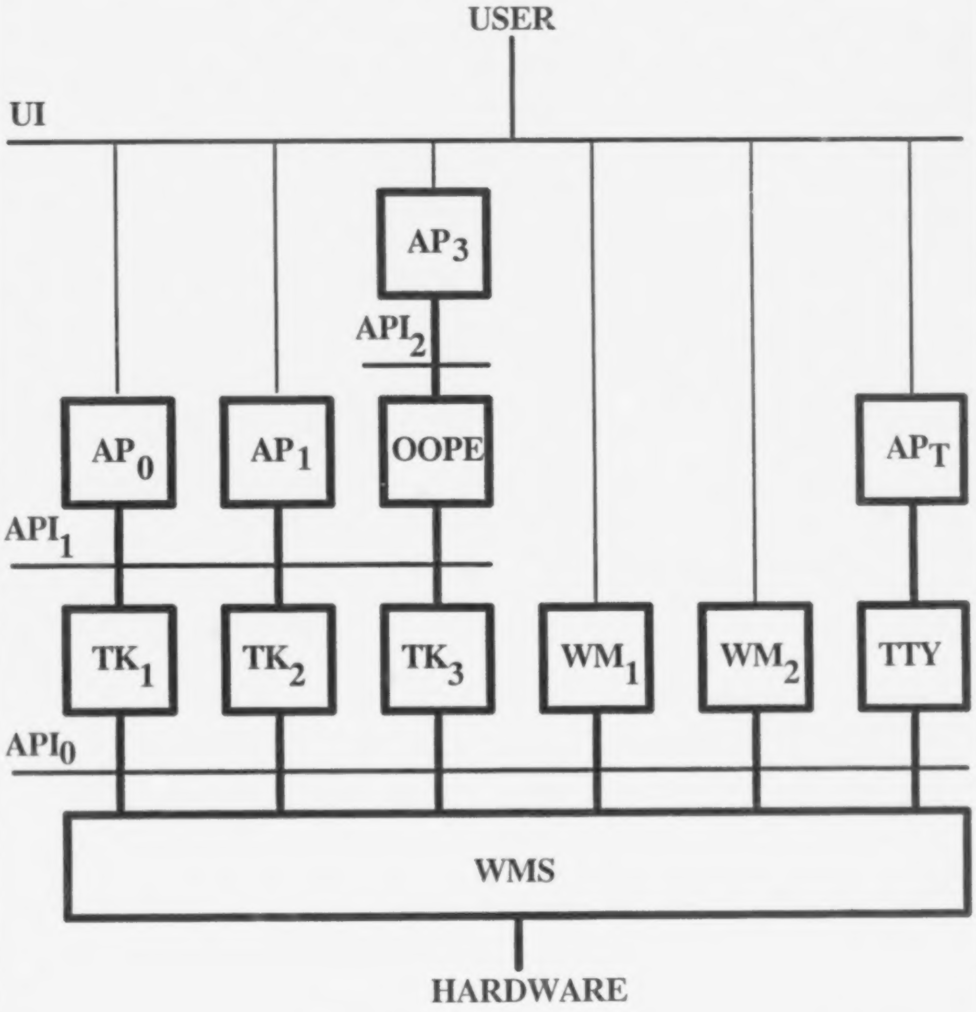


Figure 1.

396

SCRIPTWRITER II: FROM VAPORWARE TO FREEWARE

Presenter: Ellis Horowitz *University of Southern California*

SCriptWriter is a software system that can be used to create dazzling graphical applications on the IBM PC family of computers. It is anticipated that educators will use it to create computer–based instructional software. In addition, it is extremely simple to create slide shows in SCriptWriter, as well as marketing and storyboard presentations. Also, the system can interface with other PC–software. Thus SCriptWriter's applicability extends significantly beyond the educational world.

The SCriptWriter system was initially presented to the AEP schools at the first AEP conference. The reader is advised to see the conference proceedings of 1985 for a more complete discussion of the system. At the time that presentation was made, some of the components were completed while others were not. I am pleased to announce that SCriptWriter has been brought to the stage of a first release. Moreover I believe that it is the first software package that has been announced as being available to the AEP schools, free of charge.

In the next section I will give a brief overview of SCriptWriter. Then, in the following section I will talk about the system in its first release form.

SCRIPTWRITER OVERVIEW

There are four major subsystems within SCriptWriter named Graphit, FED, TED, and IQ.

Graphit is a full featured graphics editor. It includes the pre–defined shapes: circle, box, ellipse, splines, rays and grid. It has the ability to free draw using either the keyboard, mouse or digitizer stylus. Functions keys are used to control color. There is a complete set of editing functions including: paste, copy, move, blend, erase, and even a fatbits feature as introduced in MacPaint. One may also type characters by first selecting a font, size and angle. A cursor appears and you may type whatever you wish.

FED stands for Font Editor. It is provided for people who do not find the dozen or so fonts that are already included to be sufficient. Using FED one may define your own character set. There are two distinct types of fonts: bit fonts and stroke fonts. The former is created by filling dots within a grid. The latter are formed using a set of straight lines and polygons. The latter may be filled.

TED is a full screen text editor. It is used to create programs that control the manner in which forms are displayed on the screen. TED makes liberal use of the numeric key pad. It also implements a more succinct set of editing commands that are compatible with the Emacs editor.

IQ is the SCriptWriter programming language. It is a traditional language in that it works by assigning values to variables, whose type may be either integer, real or string. There is the usual set of control structures such as if–then–else, case, while, repeat–until and for statements. In addition to procedures, IQ also contains coroutines. This is the ability to have several procedures whose execution is not yet completed. Execution moves back and forth between them, producing on the screen a sense of parallel motion. The basic structuring unit of IQ is the Player. A Player is a collection of procedures and local data, and is in effect an abstract data type.

SCRIPTWRITER – RELEASE 1.0

The system is delivered on 3 diskettes and includes a manual. It can be installed on an XT, though we recommend an AT for authoring. The entire system occupies approximately 600K bytes on your hard disk. When shipped SCriptWriter has support for a Summagraphics digitizer and a Mouse Systems mouse. There are versions of the major subsystems including TED, FED, Graphit, and IQ. At present there is no VISION interface, (VISION is a system designed for interactive video applications), but that is expected in the second release.

One of the serious questions is the maximum size of an application. This is highly dependent upon the number and type of forms that are used and the amount of lines that are written. The average size of a form is 4K bytes. In general SCriptWriter can handle applications with 50–100 forms. A system that swaps forms in and out of memory is anticipated for a later release. The current speed of IQ is about the speed of Advanced Basic.

APPLICATIONS

What have we built so far in ScriptWriter? We have built several applications. Firstly, we have created many different slide shows. In fact there is a demonstration slide show that is included with each copy of the system. All that is involved in creating a slide show is to create the screens in Graphit and to place the names of the forms in the cast file. Then one may use the Cast View command to view the slides and one may transfer the screen images to 35 mm slides by using a Polaroid palette or similar device.

The next step up in complexity is to have images appear on the screen in dramatic ways. One may have an image: push up, push down, push left, push right, weave, dissolve, explode, replace left, replace right, replace up, replace down. Experience shows that such movement captures the viewer and makes for effective presentations. In SCriptWriter it is possible to have any size image appear in this way, and so one may easily add this simple form of animation to your presentation. This is done by writing an IQ program that consists of a single player, the director. His lines ask for each player to appear, in turn, and designate the option they are to use.

A more complex form of SCriptWriter application is one in which the viewer is queried and the program has control over the action. Now the play may consist of many players who have lines to say. The viewer is also considered a player. He is asked a question and the keyboard is monitored for his response. His input may be assigned to a string variable and then analyzed for specific elements. Sample applications we have built are: a typing tutor with arcade style action, a program that explains binary tree traversals, a program that illuminates the Quicksort algorithm, an adventure game in the style of Dungeons and Dragons.

Following is a sample IQ program to give you an idea of what the language looks like. This program assumes that there are 4 players named: intro, menu, endscrn, clown, trapeze, liontamer, and director. The program presents a menu to the viewer and he selects one of the actions to appear and then the program will terminate.

```
player director
entry start                  /* program execution begins here */
  integer inp                /* integer variable declarations */
  string arg                 /* string variable declarations */
  stage(1)                   /* make color monitor active */
  appear("intro")            /*the intro screen appears */
  wait(3)                    /* pause 3 seconds */
  appear("menu")             /* a menu of options appears */
  repeat                          /* wait for keyboard to be struck */
   inp = onekey(arg)
  until inp <> 0                  /* wait until keyboard is struck */
  case arg of                     /* there are three options */
  ”a” : { call(”clown” ”dance”) }       /* invoke clown to dance */
  ”b” : { call(”trapeze”, ”swing”) }    /* invoke trapeze artist */
  ”c” : { call(”liontamer”, ”howl”) }   /* invoke liontamer */
  others : {}                     /* this is an error option */
  endcase
  appear(”endscrn”)                     /* final screen appears */
  wait(3)
endentry
endplayer
```

THE CHALLENGE OF QUALITATIVE EVALUATION

Presenter: Wayne Danielson *University of Texas at Austin*

As a young man, I was one of a number of academics who helped change research in mass communication from its largely qualitative emphasis to its now largely quantitative emphasis. All my life I have been regarded as a strong advocate of quantitative methods.

It is somewhat strange, therefore, for me to come before you to speak on qualitative approaches to evaluating the educational outcomes of microcomputers in university teaching and research. As a matter of fact, in running our AEP project at the University of Texas at AUSTIN, we do a large amount of quantitative evaluation.

Faculty members propose projects following a fairly strict outline. Before each screening meeting, our committee members read all proposals and rate them using a five–point scale as shown in Table 1. We have kept all ratings since the beginning of the project in April 1984 and each new project's score is measured against all those have gone before as shown in Table 2.

However, the average quantitative ratings given proposals are only loosely predictive of whether the project will be approved by the committee. In the committee's debates, a single member who feels strongly for or against a proposal can often influence the feeling of the group as a whole. At last month's meeting, for example, a proposal that earned a disappointing 3.5 rating eventually won approval, and, I think, the decision was correct.

When a project is about six months old, its director fills out the first of a series of self–evaluations. Again, the form is largely quantitative. The director's scores are displayed against self–evaluation averages for the university as a whole. (See Table 3.) The procedure helps the staff spot projects that need help, but frankly, we usually pay more attention to the brief written comments the directors make. These comments offer many opportunities for making judgments about the director's level of expertise in the field, his or her probable ability to finish the project successfully, etc.

After about nine months, the project is visited by one or more staff members who fill out a largely quantitative evaluation form. Again, these forms are useful in a general way for keeping score on the overall progress of the grant, but those of us who review the reports pay far more attention to the hastily jotted down notes of the visitor. It is there that we seem to gain more real insight into the project and the problems it faces.

WHAT IS GOING ON HERE?

We've taken a largely quantitative approach. And I suspect that we do our quantitative work about as well as most. Yet we seem to make our most important judgments based not on quantitative data but on highly qualitative data —

* An idea or phrase that strikes a committee member as innovative or original.

* A combination of words in a self-evaluation report that identify a chief investigator sadly out of touch with his project.

* A minor note jotted down on a visitor's note pad that reveals a high level of student morale, convincing evidence that the project will succeed.

Is our project unusual in this? I don't think so. I've served on university committees and screening boards for nearly 30 years now, and almost all of them operate the same way. The members take the quantitative information, study it with serious expressions, and then set it aside and get down to business — business that is almost always qualitative in nature.

It is enough to make a behavioral scientist cry.

And yet, I have reluctantly decided, there is some wisdom here. Lord Kelvin's dictum that knowledge that cannot be expressed in numbers is "of meager and unsatisfactory kind" is simply incorrect. Human beings can speak and write as well as count. As a species we have a wonderful way with words. We can do remarkable sensitive things with them.

Not long ago I wrote an aid to poets program for the IBM PC. It's called "The Passionate Poet." It takes a simple idea provided by the poet — one line, seven or eight words — and some semantically parallel phrases. (See Figure 1.) Given some simple grammatical transformations, the program begins to ring changes on the original idea. The fascinating thing to me, aside from the fact that it is my poetry, of course, is that although the words are few and the grammatical possibilities are limited, each change is almost a new thing, each poem creates its own special and particular meaning. These poems make us smile, gasp, shake our heads, wonder. The poetry program shows how, with sixty or seventy words, we can create whole worlds of meaning. As Carl Sagan might put it, with sixty or seventy words we can create "billions and billions" of poems. We can share an infinity of meanings so precise that they put numbers to shame.

This digression is intended to show you that I am more relaxed about words now than I was as a young man. I still like numbers, but I see no reason to abandon a large part of our best human abilities simply to gain the minor communicative advantages afforded by restricting ourselves to that set of symbols and relationships we call mathematics. Often we can tell more about what is going on simply by carefully reading each other's writing and by attentively listening to each other's talk.

I guess all this is to say that I am less impressed with studies that show that teaching with computers produces greater achievement levels or faster learning rates. I am more interested in studies that try to get at new ideas, new meanings associated with teaching and research involving computers. Let me give a few examples.

It is important that microcomputers teach in private. Privacy is not always good, but it is mainly good. Students don't have to ask questions in public and display their ignorance to a scornful world. They have a private tutor who never mocks or scorns, who is unbelievable patient. "Did I spell recieve right?" the student asks his word processor. "Not quite," his patient tutor replies, "Remember, it's 'i' before e except after 'c.'" Students must go public eventually, of course, but in their own time, when they are ready.

Closely related to privacy, I think, is the fact that the microcomputer often becomes a kind of intellectual companion to the student. A fascinating theme of science fiction is humanity's encounter with alien intelligence. Most of us won't meet any creatures from the stars in our lifetimes, but most of us will meet and get to know the strange minds that reside in our personal computers. They are weird minds. They are like ours and not like ours. They are puzzling, irritating, compulsive little monsters. We tend to invest them with human characteristics. They are our R2D2 pals on an intellectual adventure. The fact that they are regarded as companions or friends is somehow significant.

If microcomputers are intellectual companions or chums, there is no question about who the leader is in the relationship. We are. I don't want to get into whips or chains here, but dominance is important in any interaction involving human beings. In ancient Greece and Rome, teachers were slaves. Somewhere in the Dark Ages, things got turned around and teachers became masters and students became masters. In England and America, we used to call teachers schoolmasters, and imposing, threatening slaves they were, too. The personal computer reverses the relationship, returning the student to the dominant role, The computer is the patient companion servant, ever ready to guide and direct, but satisfyingly under master's control. If the computer teacher ever gets really out of line, the student can always flip the switch.

Finally, as a teacher, the computer is an expert at nonverbal communication. I have emphasized words in this talk, and most of what we learn, I suppose, we learn from words and numbers. But pictures can help immensely. A chart or graph can show a relationship in seconds that would take hours to figure out from the numbers. Maps can sort out and clarify. Colors and shapes can explain and keep track of extra dimensions of meaning. The fact that nonverbals can be computed -- can respond dynamically to student needs -- makes a qualitative difference in the teaching situation, a difference that cannot be duplicated today in most conventional classrooms and laboratories.

I think you can see from these examples the kinds of approaches to evaluation that increasingly interest me. I am looking for abstractions and generalizations that go beyond the easily testable and that, like poetry, reveal new and interesting meanings. Are these meanings true, verifiable? Perhaps not. But perhaps a few, like good poems, will endure to nourish future generations.

TABLE 1. QUEST COMMITTEE MEMBER RATINGS OF NINE PROPOSAL PROJECTS

	PROJECT								
RATER	375	376	379	380	381	383	377	378	382
A	4	3	2	2	4	2	2	3	3
B	3	2	1	2	3	2	1	2	2
C	2	2	1	2	2	2	2	2	5
D	4	4	3	3	4	3	2	3	2
E	5	3	1	1	4	2	1	1	5
F	3	1	2	2	2	2	2	1	2
mean	3.5	2.5	1.7	2.0	3.2	2.2	1.7	2.0	3.2

PROJECT	PROPOSER	DEPARTMENT	MACHINES	MEAN
375		Management	2	3.5
376		English	1	2.5
379		Speech Comm.	1	1.7
380		English	1	2.0
381		Law	12	3.2
383		Astron.	1	2.2
377		Elec. Eng.	1	1.7
378		Anthro.	2	2.0
382		Anthro.	2	3.2

Table 2. NINE CURRENT PROPOSALS COMPARED WITH MEANS OF APPROVED (A) AND DENIED (D) PROPOSALS

```
                                         APPROVED PERCENTILE

------------------------------------------------------------------
375   ..................A.......D....* 3.5           1
                        A      D
376   ...................A..* 2.5                    24
                         A     D
379   ...............* 1.7     D                     86
                         A     D
380   .................* 2    D                      69
                         A     D
381   ...................A.......D....* 3.2          1
                         A     D
383   ................*2.2     D                     50
                         A     D
377   ...............* 1.7     D                     86
                         A     D
378   .................* 2     D                     69
                         A     D
382   ..................A.......D....* 3.2           1
                         A     D
------------------------------------------------------------------
 0                       2.2   3          5
```

TABLE 3. SELF EVALUATIONS FROM 107 QUEST PROJECTS AS OF 04-02-1986

```
        HARDWARE SATISFACTION %'S ( AVERAGE % = 89 )
%   42                                                   H
    48                                                   H
O   34                                                   H
    30                                                   H
    26                            H                      H
P   22                            H                      H
R   18                            H          H           H
O   14                            H          H           H
J   10                            H          H           H
    6                             H   H      H      H    H
    2                             H   H  H   H  H   H    H
-----------------------------------------------------------------
0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100
                    PERCENT REPORTED
```

```
        SOFTWARE SATISFACTION %'S ( AVERAGE % = 79 )
%   31                                                  S
    28                                                  S
O   25                                                  S
F   22                            S                     S
    19                            S                     S
P   16                            S                     S
R   13                            S                S    S
O   10                 S          S                S    S
J   7         S        S          S    S     S     S    S
    4         S        S          S    S     S     S    S
    1         S        S   S S    S    S     S     S S  S

-----------------------------------------------------------------
0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100
                    PERCENT REPORTED
```

```
               COMPLETION %'S ( AVERAGE % = 47 )
%  2                                                   C
   18                                                  C
O  16                          C                   C
F  14                          C                   C                   C
   12                          C                   C                   C
P  10                          C           C       C                   C
R  8                       C   C           C       C                   C
O  6                       C   C           C       C       C           C
J  4                   C       C   C   C       C   C   C       C   C           C
   2                   C       C   C   C   C   C   C   C   C   C   C       C       C   C       C

--------------------------------------------------------------------------------------------

           0   5  10  15  20  25  30  35  40  45  50  55  60  65  70  75  80  85  90  95 100
                                   PERCENT REPORTED

AVERAGE NUMBER UNDERGRADUATE STUDENTS =  20
AVERAGE NUMBER GRADUATE STUDENTS =  5
AVEARGE NUMBER FACULTY =  2
```

FIGURE 1. POEMS PRODUCED BY THE PASSIONATE POET

Apple
boys
sigh
for her
long
beds.

Bad
brides
rage
in their
winter
kitchens.

Green
fires
pulse,
sink
in his
silent
lamps.

Raucously
bare
brides
cry.

Across Denver
green
pines
bloom,
pulse.

TOOLS FOR LEARNING AT THE INTERACTIVE COMPUTER GRAPHICS LABORATORY, PRINCETON UNIVERSITY

Presenter: Kirk Alexander *Princeton University*

The Interactive Computer Graphics Lab at Princeton has been supporting and developing a number of graphics applications for over ten years. While a number of major projects have come and gone (like an online NC milling machine), the following activities have proved to be areas of continuing need and interest:

* Transportation Network Graphics
* Graphics Object Generation & Editing (figures)
* X-Y Data Plotting (graphs)
* Simple graphics software tools for individual applications
* Printing Integrated Text and Graphics (JANUS with plots from a variety of sources)

The transportation activity as well as a couple of other activities such as statistical graphics and image processing have even been reborn outside the university as commercial software ventures.

CURRENT USER NEEDS

Today many of the activities once supported by the lab are available in many other forms as various micro-computer software offerings. However the lab now is trying to broaden its scope and assist users around the campus with those graphics problems that are either too expensive or too difficult to be managed by the more common PC configurations. Among the most frequent needs expressed by university users are

* Quality Color Hardcopy at Low User Cost (final draft)
* "Real-time" Animation
  - if not easily computed - at least easily recorded
  - Engineering, Architecture, Visual Arts
* Display of Multi-dimensional Matrix Data

- multiple instances (e.g. rainstorm)
- multiple cross sections

CURRENT ICGL ACTIVITY

In response to these needs the following areas represent the current areas of major ICGL activity:

- Simplifying Presentation of 3D Data
- Simplifying Display of Image (matrix) Data
  - This area is being addressed with two major activities...use of the Decision Images PC based Image Processing System and local development on the IBM 5080 Graphics System. The latter includes
    - 5080 Image Cursor
    - 5080 Grid with Polygon Fill
    - 5080 Image Placement
- Hardcopy

Here the lab is investigating devices than can provide reasonably high resolution images with faithful color reproduction. Both film recording devices and ink-jet printers are being investigated.

- Animated Sequences of Data Display

A number of users have programs that generate output at different time steps or at different locations in space. We are working with or building data structures that can simplify the display process of such data. For the former case the concept hinges around 'page flipping' or incremental solution sequences. For the second, we are building object trajectories that define both the viewer's trajectory through time and the object's trajectory.

DIFFICULT AREAS YET TO CONQUER

Still, there are many difficult areas yet to conquer. While there are many software packages available to help with any one problem, bringing them altogether as one large tool-box is still beyond the immediate horizon. The areas we see as most challenging include:

- Bringing together applications, graphics and user-interfaces
- Understanding Spatial Concepts and Design
- Graphical Display of Algorithm Development

- Display of multi–dimensional data
- Integrating Decision Assist software with graphical data base display
- Integrating Pixel and Vector technologies
- Bringing existing tools into new problem domains
- 3D and Shading Techniques (any multi–dimensional data analysis)
- Image Processing Techniques (Fluid analysis, Medical)
- Intelligent User Design and Control of Animation
- Massive Data Manipulation
- Data Communications

WHAT WE HAVE LEARNED

In the form of an outline summary there have been a few points that have reappeared again and again in our experience:

- Multiple Ways of Looking at Data are REQUIRED.

- REPORTING – People Have to Do It.

- Good tools backed by assistants who know how to use them get used a lot and make people VERY productive.

- Productive people achieve more insights into their own work.

- Tool developers and maintainers NEED "big" scale projects to provide a continuum that holds interest and provides development continuity.

- Student work plugs into a continuum but does not provide it.

- A concentrated specialty (e.g. Graphics) knowledge pool yields better tools and provides a better resource than just isolated exploration.

- ICGL "buy–in" staff policy allows both resource–pool and departmental individuality benefits.

In short, the boon of distributed computing and personal workstations has brought an incredible amount of function to an ever growing community of users. However, the personal computers of today are themselves growing faster than any one person can keep up with. The Interactive Computer Graphics Lab at Princeton remains interested in trying to understand what applications do not currently make a good fit and why. We will continue to explore both hardware and software paths that allow us to solve the greatest number of user needs with the fewest discordant components. It means maintaining the difficult balance with cutting edge technology that is not compatible with anything and the powerful wealth of current tools that may not yield to new concepts.

TOWARDS AN INTEGRATED DESIGN SYSTEM FOR THE IBM AT

Presenter: Gerhard Schmitt *Carnegie–Mellon University*

Coauthors: Gerhard Schmitt *Carnegie–Mellon University*
Jean–Christophe Robert *Carnegie–Mellon University*

INTRODUCTION

The effective use of CAD in design education on affordable hardware is still suffering from missing links between available software. Although some packages are extremely powerful in their domain such as drafting, databases and spreadsheets, no single program can presently address the entire range of quantitative, qualitative and visual decisions facing the designer. We therefore started to link several existing packages and to add missing features to create a more complete, flexible system that allows various additions.

The IBM AT was chosen as the implementation environment. Reasons are the rich existing software base and its widespread use in education and business.

The hardware and software requirements are:

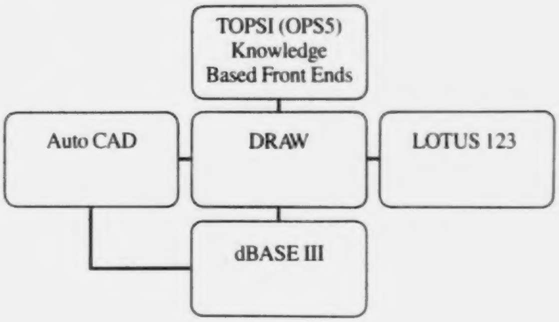
IBM AT 512 K
Hard disk
Virtual disk (recommended)
Professional Graphics Adapter and Display
MouseSystems Mouse
Plotter

GKS, C–Bindings
VDI
Lattice C Compiler
TOPSI

The system developed so far consists of:

- A graphics program, DRAW
- Interfaces to Lotus 123 and AutoCAD
- Rule–based front ends

Interfaces with dBase III are under development.



The system allows a large number of operation modes. For example, a design can be created using one of the expert front ends, displayed and visually edited with DRAW, dimensioned and supplied with attributes in AutoCAD, and analyzed or stored in dBase III. AutoCAD drawings can be displayed as perspectives with DRAW. Graphics information entered in DRAW can be used in Lotus 123 for simple cost or energy analyses.

The following is a more detailed description of the different parts of the system.

DRAW

INTRODUCTION

DRAW is a graphics program written in C and based on the IBM Graphical Kernel System. It was developed to comply with the two following goals:

It will be used for educational purposes. It must thus feature the essential facilities of a graphics package in a user-friendly environment.

It must have interface capabilities with several other existing programs.

PROGRAM DESCRIPTION

According to the above goals, the characteristics of the program were determined: selection of graphical functions, user interface, and data format.

User Interface

The user interface is composed of nested menus displayed on the right-hand side of the screen, in which the user picks the item of his choice. The graphic input is performed by positioning the screen cursor. The use of a mouse is strongly recommended since it enables very fast interactions. In addition, messages displayed at the bottom of the screen tell the user what action he has to perform.

Graphic Primitives

The program allows three basic graphic primitives: lines and polygons, both defined by a set of vertices, and arcs (or pies), defined by center point, radius and angle.

Attributes

The icon driven menus allow the user to select color and type attributes for each primitive. Line-type can be chosen among six patterns, and polygons and pies accept eight fill-patterns: hollow, solid and several hatch styles.

Axis

The model space is described by a Cartesian coordinate system. The three directions X,Y,Z are represented on the screen.

Display Window

The viewing of the model results from a projection on a plane. The specification of the plane is performed by selecting a type of view (plane, perspective..). The screen is a window that displays a portion of this plane. Several functions provide a means of choosing which portion of the model will be displayed, by setting its location on the plane and its size. This can be performed by typing in the coordinates of the window's center and the length of its side, or by selecting an area on the screen. Zooming effects can thus be obtained. It is also possible to translate the screen-window on the projection plane. Another option computes the size and location of the window in order to display the whole model, centered on the screen.

Extrusions

DRAW handles also 3D–extrusions of plane figures. Wire frame 3D objects can thus be represented. They are obtained by specifying an elevation and a height, which determine the Z–related bottom and top coordinates.

Views

The program features X–Y plan view, X–Z and Y–Z elevations, axonometrics and perspectives. For axonometric views, the user has to enter the view direction vector coordinates. For perspectives, DRAW requests the coordinates of the viewpoint and of the eye–target direction. However, three predefined one–, two– and three–point perspective views are provided. For these, only the viewpoint is requested. For every type of view, all the display control functions (WINDOW, CENTER, MOVE...) are available.

Data–File

Each time a graphic primitive is created, the corresponding data are recorded in a file. The file format was designed in order to simplify as much as possible data exchanges with other programs, such as AutoCAD or Lotus 123. Therefore data are stored in ASCII format and are arranged into lines that define either an attribute or a primitive.

Each line contains a character that shows what the line refers to, followed by numeric data.

- Color: C *color*

- Line type or fill pattern: *T type*

- Line: L *elevation height* x_1 y_1 ... *xn yn*
- where *xi, yi* are the coordinates of the line vertices.

- Polygon: **P** *elevation height x1 y1 ... xn yn*

- Arc: **c** *elevation height x center y center radius angle1 angle1*

417

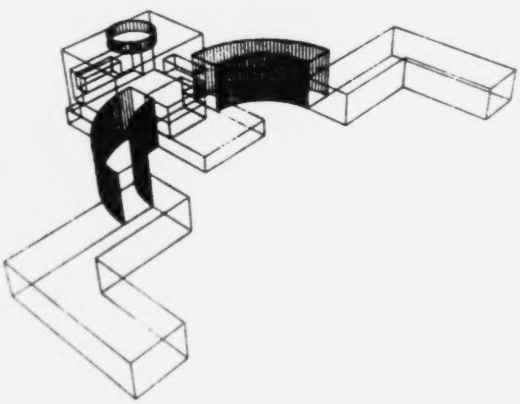
A color or type specification affects all the following primitives until another attribute specification is encountered.

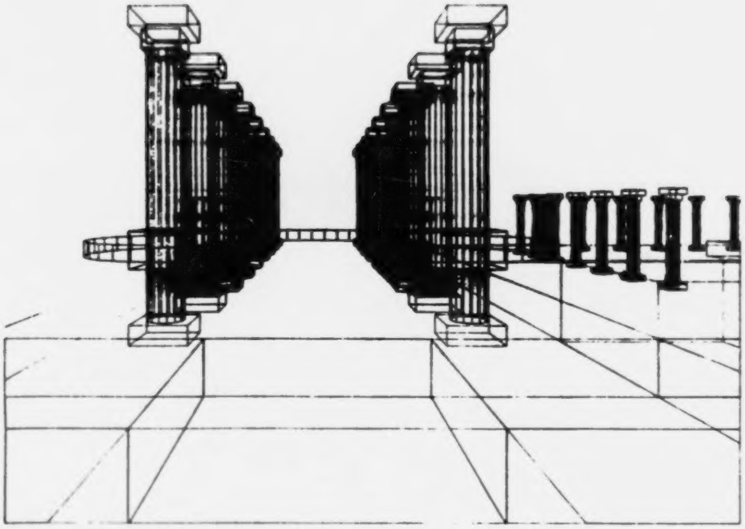
Editing

The user can either create a model or read and edit an existing one. The HISTORY function recapitulates all the successive element creations and provides a way of deleting every unwanted primitive.

FURTHER DEVELOPMENTS

The program, in its present form, features a limited set of graphic functions. This helps to satisfy the initial goal of user–friendliness. However, some further developments would enhance the capabilities of DRAW. In preparation are a hidden lines removal option, and an extension of the editing facilities. Finally, other primitives could be added, such as polygons not contained in an X–Y plane.





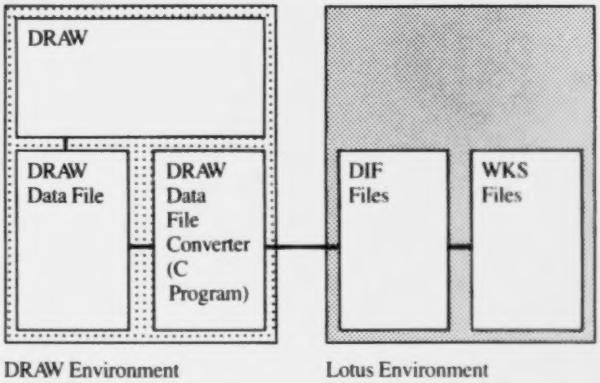
DRAW - LOTUS 123

INTRODUCTION

The very intelligible format of data described above enables the use of a DRAW file for edition of data or other purposes. It seems therefore very useful to enable the transfer of such a file into a spreadsheet environment. An interface between DRAW and Lotus 123 was thus developed.

FORMAT CONVERSION

The interface performs the data conversion between the two file formats considered: DRAW and Lotus Worksheet (.WKS). For both directions the file that has to be converted must be first transformed into an intermediate format (.DIF), required by Lotus 123. The conversion between .WKS and .DIF formats is provided by the "TRANSLAT" Lotus command. Two C programs were developed to achieve the conversion between .DIF and DRAW formats. They merely read the initial file and rewrite it in the final format.



INTERFACE BODY

All the successive format conversion operations are called by a batch program. This program also includes key-stroke simulations in order to avoid any unnecessary typing, especially when running Lotus.

From within DRAW, the batch program is called by selecting the appropriate item out of a menu. Then the current DRAW data file is converted in two stages into a .WKS file. This file is then displayed in the Lotus environment, and Lotus specific macros can be applied. When leaving Lotus, the reverse conversion is performed before going back to DRAW.

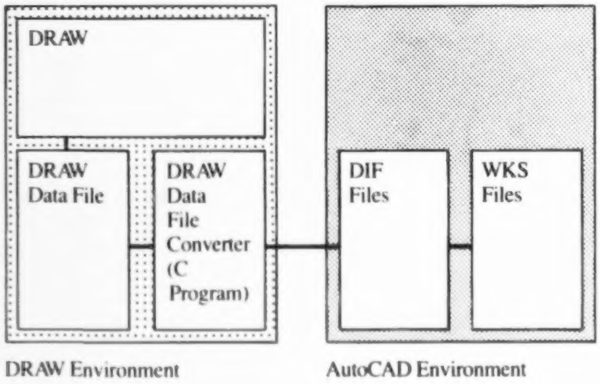
DRAW - AUTOCAD

INTRODUCTION

DRAW features a limited set of graphic functions in comparison to a commercial graphics package such as AutoCAD. However, some actions can be performed in DRAW that are not possible in AutoCAD (perspective views, for instance). An interface was therefore developed in order to enable the transfer of models between DRAW and AutoCAD and to take advantage of all their facilities.

DATA TRANSFER

Data are transferred by first converting the appropriate file into an intermediate format imposed by AutoCAD (.DXF). AutoCAD functions perform the conversion between AutoCAD drawing files (.DWG) and .DXF files. Two interface programs were developed for the conversion between .DXF and DRAW formats.



SWITCHING FROM ONE PROGRAM TO THE OTHER

Similar to the interface described above, all the transfer operations are called by a batch program. It allows the user to switch from one program to the other without extra typing. The switch command is called by menu-selection.

DRAW - KBES

INTRODUCTION

The motivation of improving the convenience of DRAW led to the development of new user interfaces. They were implemented in TOPSI, the IBM AT version of OPS5, a production system developed at Carnegie–Mellon University. OPS5 is a rule–based language for programming expert systems. It consists of production rules made of Left Hand Side (LHS) conditions and Right Hand Side (RHS) actions. It was chosen mainly for the flexibility it allows. Two "expert" front ends were thus developed. They both accept input from the user and then display the resulting model in DRAW.

FILE CREATION

Once the user has performed the input operations through the expert front end, the resulting DRAW data file must be created. This operation is accomplished by C procedures annexed to TOPSI and called by TOPSI rules. The resulting file is stored in a virtual disk in order to improve the speed of the process.

EXAMPLE 1: CUSTOMIZED PRIMITIVES GENERATOR

This program enables the user to select graphic primitives from a set of primitives that can be easily expanded by adding new rules.

User Interface

The front end provides the user with a set of graphic primitives. The user chooses a primitive, and adds related numerical information requested by the program. After selecting the "DISPLAY" option, the resulting drawing is displayed in DRAW. At this stage all the DRAW facilities can be used to edit the model or to choose a different view. When leaving DRAW, the user returns to the TOPSI front end and can keep on adding elements to the model before asking for another display.

Generation of Primitives

An important aspect of this program is that it improves the capabilities of DRAW by allowing new graphic primitives. The creation of a new primitive is accomplished by adding three rules to the front end: one to add an item to the selection menu, a second one that performs the numerical data input, and a third one to define the resulting DRAW output. In addition, a primitive can be composed of several other existing primitives. The procedure is therefore very flexible.

The first basic primitives implemented were lines, squares and arcs (including circles). Then columns were created, composed of three squares and one circle. Eventually rows of columns were added, by putting columns side by side. Additional implementations include cones, pyramids, and domes.

This first example, featuring a very simple user interface, shows how an uncomplicated though powerful expert front end can extend the capabilities of DRAW.

EXAMPLE 2: RELIGIOUS BUILDINGS GENERATOR

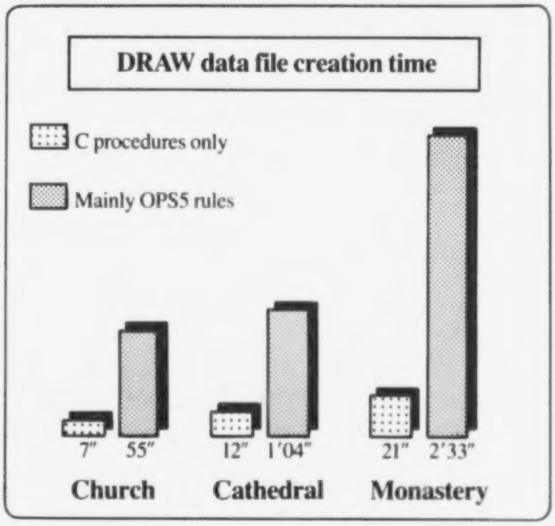
This program aims at the generation of religious buildings, according to information entered by the user. It was developed to explore the representation and interaction of qualitative and quantitative symbols in the same program.

"Intelligent" User Interface

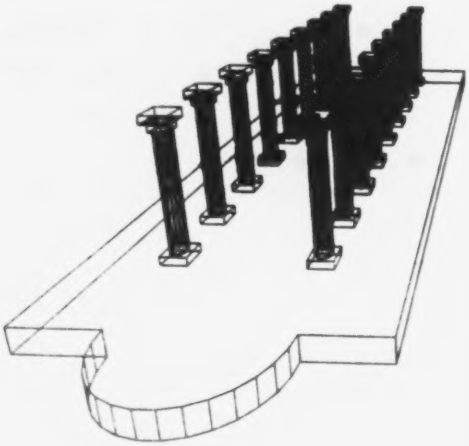
Unlike in the previous example, the user interface is designed to input information not directly graphic-related. A number of choices are presented to the user in form of short menus. Choices include the building type and location, the kind of religious order and the financial situation of the builders. If these parameters are sufficient to characterize a building adequately, the program will call DRAW and display it. The user can at this moment add extra elements to the building or try different perspective views. When he quits DRAW, he is requested for information to generate another building.

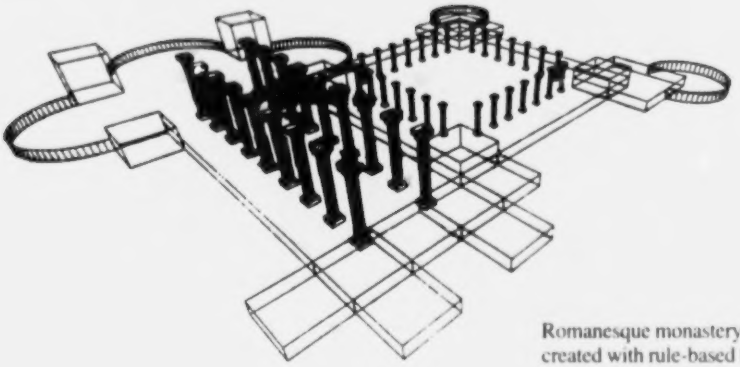
Religious Buildings

The religious buildings are examples of complex primitives defined in a similar way as in the previous program. However, in order to make the program more efficient, the characteristics of the buildings are no longer recorded in TOPSI rules, but in C procedures. This decreases the flexibility of the program, but saves computation time.



This second program builds on the same basic ideas as the previous one, but features a much more elaborate user interface. It is a first step towards graphic output of more complex expert systems.





Romanesque monastery
created with rule-based front end
displayed with DRAW
(3 point perspective)

GRAPHIC KNOWLEDGE ACQUISITION

INTRODUCTION

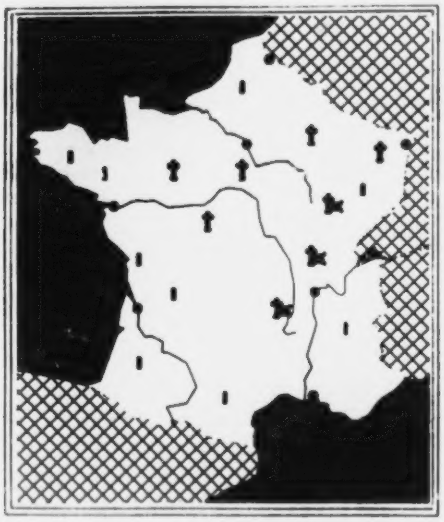
While running a TOPSI program, the user generally enters information by typing in answers to questions generated by the expert system. With the graphic knowledge acquisition program some input can now be performed graphically. The new concept implemented here is the generation of TOPSI rules according to graphic input of symbols into a site.

SITE AND SYMBOLS REPRESENTATION

When called by a TOPSI rule, the knowledge acquisition module displays several icons in the menu area of the screen and a site in which icons have to be inserted. The site is divided into several regions. The user performs the input by picking an icon and by inserting it into the appropriate area. The boundaries of the regions can be displayed or removed. Each icon and the site itself are recorded in regular DRAW files, thus enabling an easy creation and editing of the graphic symbols to be used.

GENERATION OF TOPSI RULES

Once the symbols are located in the site, the program generates TOPSI rules containing the associated information. One rule is created for each region, and records the quantity of symbols of each type entered in the region. The newly created rules are then added to the existing set of rules, and the expert system continues execution.



EXAMPLE: MEDIEVAL ARCHITECTURE CONSULTANT

This prototype is an expert system that generates political and economical statements about five regions of medieval France, depending on the types and number of religious buildings encountered. Therefore the site displayed is a map of France and the icons represent three religious building types: cathedral, monastery and church.

The program presented here shows an original way of adding rules to an expert system. It could be adapted to other uses, for instance building diagnostics: possible performance problems would be inserted graphically in the building plan, and a solution would then be suggested.

ACKNOWLEDGEMENTS

The authors would like to thank Chia–Ming Chen and Ann Boylan for their support and contributions.

YOU CAN GET THERE FROM HERE: GUIDELINES FOR THE EVALUATION OF INSTRUCTIONAL COMPUTING

Presenter: **Michelle Comeaux** *University of Wisconsin, Madison*

Coauthors: **Michelle Comeaux** *University of Wisconsin, Madison*

Clifford Gillman *University of Wisconsin, Madison*

"Would you tell me, please, which way I ought to go from here?" [asked Alice]

"That depends a good deal on where you want to get to," said the Cat.
(from Lewis Carroll's Alice in Wonderland)

Few designers of instructional software would deny that evaluation is an important component in the process of creating microcomputer software for use in the classroom or laboratory. Yet frequently it is the component that receives the least amoun

1.The attitude that since it has been "proven" that instruction with microcomputers is at least as effective as traditional teaching methods, no further testing of its impact on learning needs to be done.

2.The lack of relevant research and subsequent publication on methodology designed specifically for evaluation of instructional computing.

3.The belief that if students find instruction with microcomputers "fun" and "interesting" that is enough to justify its use.

4.The misconception that evaluation is something tacked on at the end of development.

It is of little dispute that computer–based instruction is as effective, and in some cases, more effective when compared to "traditional methods". Kulik, Kulik, and Cohen (1980) in a statistical review of 59 independent evaluations of computer–based college teaching found that "computer–based instruction made small but significant contributions to the course achievement of college students." In addition, their meta–analysis revealed that CBI (computer–based instruction) also produced positive changes in students' attitudes toward instruction and subject matter. Similar findings regarding the effectiveness of CBI in elementary and secondary classrooms have been reported by Macken and Suppes (1976) and Thomas (1978), respectively. As Kearsley, Hunter, and Seidel (1983) conclude in their review of CBI projects, "There is ample evidence that computers can make instruction more efficient or effective. "

Such findings, however, are only one piece of the computer–based instruction puzzle. Roblyer (1985) states well the case for not over–relying on studies pitting the computer versus "traditional methods" for evaluative data:

> These results, while helpful in indicating usefulness in a general way, do not add substantially to our knowledge of specific computer–related variables which enhance or impede learning. Statistical methods such as meta–analysis have given us insight into the meaning of trends in research, but these analyses cannot supply specific information to guide the development or implementation of computer–based materials.

Likewise, Salomon and Gardner (1986) call for evaluation that looks beyond a "successful" or "not successful" orientation and concentrates instead on which variables make CBI effective and for whom.

Evaluation such as that called for by Roblyer and Salomon and Gardner necessitates the employment of evaluation methods throughout the development of instructional courseware. As Lai (1984) notes, an evaluation mistake to be avoided is that of the "stand alone, summative, go/no–go study". Unfortunately, as Lai also notes, there is a dearth of useful and practical writings in educational literature regarding evaluation for the developer of CBI. Evaluation techniques often relied on are simple single group pre–post measures, which are highly susceptible to methodological flaws (Mueller, 1985) and reliance on verbal reports that students found CBI "fun" or "interesting". The erroneous assumption behind this latter technique is that if the student finds it "fun", he/she therefore learns more. Such is not always the case, however (see, for example, Clark, 1982). While it is hoped that the "run" component does add to the student's motivation, it is not enough to insure that effective learning has taken place. Indeed, it could be argued that once the novelty of

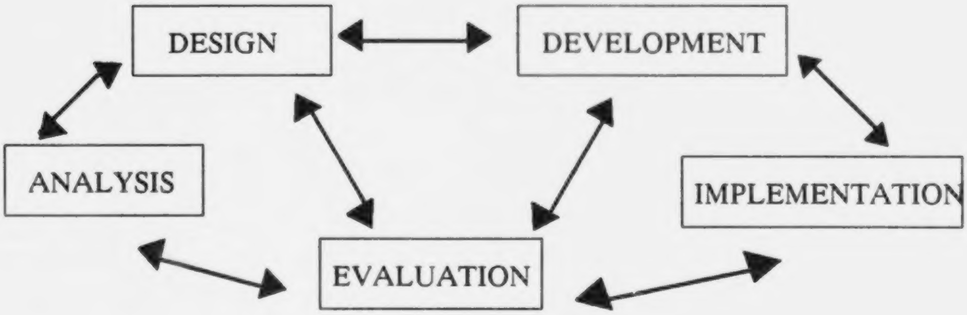
CBI wears off, CBI offers little advantage over traditional methods in that regard.

At this point the developer of instructional courseware might well be wondering, like Alice, "which way I ought to go from here". Where does evaluation fit in and how is it accomplished? The rest of this paper will address that issue and hopefully, supply some answers. As will be suggested, evaluation does not take place at the end only, but rather is an integral part of every phase in the production of instructional courseware.

THE ROLE OF EVALUATION IN INSTRUCTIONAL DESIGN

The genesis of CBI often begins with a wondering, or a "What if.." – "I wonder if there is some way I can increase my students' exposure to the spectrometer given the number of instruments in the lab?" or "What if I could model the image I have in my head of Bernoulli trials so that students could see how it looks?" Sometimes, however, a fascination with the computer comes first and then an attempt to find some way to use it. The computer then becomes a "medium in search of a message" (Lehrer, 1986) or as Jernstedt (1982) observes, "the effort to improve instruction is often technology–driven rather than education–driven." To insure effective and efficient use of the computer, the educational end result desired must be kept in mind at all times; instructional design is, to some degree, a matter of working backwards. As the Cat suggests to Alice, to figure out how to get someplace, you must first know where it is you want to go, or to extend the analogy – what you want to achieve.

Many models of instructional design exist and while labels for the various components differ, the processes are remarkably similar (see, for example, Billings, 1985; Dick & Carey, 1985). One popular model of instructional design is that developed by Kearsley (1984), specifically for use in CBI. The model consists of five phases:



(Kearsley, 1984, p.105)

The model assumes that an educational objective or goal is what is driving the entire process and it is kept in mind throughout the entire process. A brief description of activities comprising each of the five phases follows, based on Kearsley's model and similar models by Billings and Dick and Carey.

ANALYSIS: (1)Identify the resources, money, and time needed to develop the software. This phase may include writing a grant proposal, securing financial support from a department chairperson, and finding programmers. The amount of time taken to compete the project should not be be underestimated. Billings notes that one hour of CBI can require form 50 to 700 hours of development time.

(2) Confirm via a media search that suitable software does not already exist that would accomplish the specified educational objective.

(3) Analyze the learners – what prerequisite skills must they have to benefit from instruction? Are they novices or experts in the content area? Literature on problem solving suggests that novices and experts solve problems in different ways (see, for example, Larkin, McDermott, Simon, & Simon, 1980).

(4) Examine the nature of the learning outcome – is the student to learn a concept? skill? declarative information? Under what conditions should the student be expected to exhibit the learned behavior – in the laboratory?

classroom? clinical setting?

DESIGN; Select the system characteristics such as display, how information is input, control of program, mode of interaction (e.g. simulation, tutorial, drill and practice). Decisions regarding design should have a theoretical base and conditions which foster learning should be given prime consideration and inclusion. Jay (1983), Steinberg (1983), Cohen (1983), Grabowski and Aggen (1984) and Wade, Jr. (1980) all discuss findings in cognitive psychology that are relevant to the design of computer software. For example, the limits of working memory dictate that the amount of text on the screen should be controlled. Long texts should be avoided; if the screen is full of print, the amount of memory left to attend to other tasks such as the coding of information for long term storage is decreased. Also, graphics can enhance the storing of information in long term memory due to the elaboration it provides. Therefore, color should be used to code, differentiate, or highlight key concepts. When possible, ideas should be represented with pictures, graphs, maps, etc. as well as text.

DEVELOPMENT: Develop the instructional software and get it "up and running".For those designers who have neither a solid background in programming skills nor a person on the team who has programming capabilities and experience, this can easily be the most frustrating and time–consuming phase. This phase includes what Kearsley calls the "authoring process" which consists of four basic steps: (1) representation of content and logic of learning activities in some way (e.g. a flowchart, storyboard, or diagram) (2) programming of content and logic (3) debugging and testing of program and (4) production of any documentation needed such as workbooks, tapes, video (from Kearsley, 1985). Also included in this phase would be the development of tests, if they are to be used, for assessment of the student's learning.

IMPLEMENTATION: Try out the instructional courseware on the target students in the operational setting. Revise the program, both documentation and software, based on performance by the students and organizational considerations.

EVALUATION: Use a variety of techniques, both quantitative and qualitative, inorder to refine the instructional courseware created and to judge its effectiveness for the intended users. Evaluation does not come last, but rather as shown in the diagram as an activity that goes on during all the phases, from analysis through implementation. There is a continual feedback loop operating between each phase and the evaluation phase. Thus, in a sense, evaluation is really a higher order phase which the other phases must match their standards against. Evaluative data may be gathered, for example, in the analysis phase to determine what level of computer skills the students have coming into the course so as not to create a program that demands more expertise than they have (or else to build in the appropriate training steps during the design phase). During the design phase, pilot testing takes place with subject matter experts, individual students, and small groups of students. Finally, during implementation, the instructional courseware is used by the target audience and instructional effectiveness is gauged. The next two sections of this paper will discuss in greater detail the methodology used in evaluation, for as noted earlier, it is the phase most often short-changed in the production of CBI.

EVALUATION

Few researchers who write articles for scholarly journals would think of submitting a paper that is a "rough draft". Before submission, it is a common procedure to have colleagues read and critique the paper, to make revisions based on their suggestions, and then have it critiqued again. Yet frequently this step of revision is ignored in the writing and development of instructional courseware. Brown and Ellinger (1981) describe the scenario well:

> Beginning with a fair level of enthusiasm for the capabilities and options provided by a computer-based delivery system,the would-be user of computer-based instruction initiates his development effort....Following a period of time, which is almost inevitably longer than planned, the audible sigh of relief, and perhaps an unstated resolution to never become involved in such a development effort again, the developer takes the required logistical steps to make the material available for students' use. Casually, and almost unnoticed, the assumption has been made that the material is adequate and effective and is ready for general use...all too often the development effort has become so aversive that the thought of revision, if it occurs at all, is quickley banished from further consideration.

As Brown and Ellinger note, two undesirable side effects of using the program in its "rough draft" form are that (1) the material is not improved to make it more effective and (2) auxiliary materials such as a student's guide,instructor's manual, etc. are not developed. In order to provide students with instructional software of the highest quality and insure its successful use in the classroom, two types of evaluation must occur: formative and summative.

The distinction between formative and summative evaluation is less one of methodology, than of purpose (see, for example, Golas, 1983; Kandaswamy, 1980; Kearsley, 1984; and Knapper, 1981). In general, the data gathered in formative evaluation is used to revise and improve the instructional program while the data collected in summative evaluation is used to judge the overall effectiveness and value of CBI.

FORMATIVE EVALUATION

Formative evaluation is an iterative process which occurs primarily in the analysis, design, and developmental phases of the instructional design process. It is beyond the scope of this paper to discuss all the evaluation factors that should be considered in the analysis and design stages. The reader is referred to Appendix A for a listing of attributes to consider during the analysis and design stages. Generally, students are not involved in the design and analysis stages of evaluation, but rather subject area experts, instructional designers, and educational psychologists are brought in to evaluate the program regarding content, design, and pedagogical soundness.

While it certainly is advisable to include the above mentioned experts in the evaluation, it is not wise to make them the only source of feedback in the program's evaluation. For several reasons, once the program is capable of being run, students should be used to help evaluate its instructional soundness. First of all, there is some debate regarding the reliability of experts in judging the effectiveness of instructional programs. Rothkopf (1963) in a study employing "expert" multiple raters found that predictions of effectiveness of instructional materials were actually negatively correlated with observed student performance! Second, students often interact with a program in ways that cannot be predicted beforehand. Both qualitative and quantitative data can be gathered from students. Included in the former category are observations and verbal protocols; included in the latter are attitude scales, learning outcome measurements, and frame analysis.

QUALITATIVE DATA: A seemingly simplistic, but very powerful technique for assessing student interaction with the instructional software, is the combination of observational and verbal data. Used heavily in the discipline of cognitive psychology, verbal protocols are a rich source of data. The procedure is not difficult, although it can be time–consuming. The only materials needed are a tape recorder, tape, notepad for making observations, an "up and running" program, and, of course, the student. The student is asked to "think aloud" as he/she goes through the program; a tape recorder records the running commentary of the student. The interviewer must be careful not to make statements that will in any way influence the student's decisions (for an excellent discussion of this technique see Ericcson and Simon, 1980). As the student works through the program, the interviewer notes such things as the length of time needed by the student to read the screen, student raction to or awareness of feedback, signs of boredom or restlessness, etc. After the verbal protocol is completed, the tape can be transcribed and examined by the developer.

An example of a verbal protocol can be found in Appendix B. The program evaluated is one developed by Larry Boh, Michael Pitterle, and Joseph Wiederholt of the School of Pharmacy at the University of Wisconsin–Madison. The objective of the program, which is a simulation, is to improve the student's patient medication problem solving skills. One of the authors of this paper was the interviewer and has had practice collecting verbal protocols. By examining the protocol (the actual protocol was seven pages long), the developers learned that the student lacked the necessary background information to read the charts given, did not understand or use some of the commands, and did not know how to locate certain information. Such feedback led them to add a tutorial to the program, highlight various commands, and write a workbook/reference guide to accompany the software. Verbal protocols were collected from several other students and each protocol yielded more data about students' misconceptions, lack of necessary background information, or misunderstanding of directions.

Once the verbal protocol has been collected, the student should in addition be interviewed to find out his/her overall reactions to the program. The student can be asked to pinpoint any frustrations he/she experienced, to comment on what he/she liked about the program, and to make suggestions on how it can be improved. The interview should be conducted immediately after the program is finished so that the impressions are still fresh in the student's mind and not interfered with by time or conversations with other classmates who may have tried out the program. It is desirable to collect verbal protocols from several students who vary in sex, ability, motivation, and background so as to determine if such factors interact with the program's effectiveness.

QUANTITATIVE DATA: Quantitative data should be collected on a group of students, ideally 5 –10 in number. Quantitative data consists of primarily three kinds: attitude scales, learning measure outcomes, and where applicable, errors by frame. It should be stressed, however, that it is not the student that is being evaluated, but rather the program. Also care should be taken that (1) students should be representative of the intended target groups and (2) the conditions under which the trial subjects are tested should as nearly as possible resemble those conditions in which the material will actually be used (Brown and Ellinger, 1981).

Students' attitudes toward the instructional program are important to assess. It is obvious that if they find the program boring, threatening, or confusing, learning will be affected. While students should also be interviewed to find out their attitudes, it is more efficient to administer a paper and pencil or computer survey when answers to more than five questions are desired. In addition, pre–established surveys can add uniformity and specificity to the information gained. Examples of paper and pencil and computer surveys are found in Appendix C and D. In the example in Appendix D, the survey was conducted at the end of the instructional program and gave the instructors immediate feedback on how the program was regarded by the student. The computer also served, in this case, as a memory bank for the responses and as such, reduced the amount of time needed by the instructors to analyze data.

Surveys can include questions that tap such variables as understanding of the program's intent, computer anxiety/confidence, previous experience with computers, perceptions of feedback, etc. Survey results can also be analyzed to locate possible aptitude by treatment interactions. For example, do those with high computer anxiety do more poorly on the performance measures than those with low anxiety?

A second source of quantitative data are performance measures which address the question: Did the student learn what the program was designed to teach him/her? Usually it is during the development phase that measures are designed to test the objectives of the program. Part of the evaluation then is to assess how students perform on the measures, and if the performance is below a previously established range of acceptability, to determine if the fault lies in the program or the measurement instruments (i.e. do they lack construct validity?). The process of measuring the knowledge a student has acquired as a result of using the program is a complex one and measurement instruments will vary depending on the mode of interactionused for teaching and the educational objectives. For example, a multiple choice test may be suitable for measuring knowledge gained via a drill and practice program but is generally not useful for testing knowledge gained through use of a simsulation.

Appendix E lists the various modes of interaction a program designer has to choose from, gives an example of each, and lists measurement concerns for each type. The categorization scheme was derived largely from Patterson and Smith (1985) with additional description added from the work of Bell (1985), Cohen (1983), Hewson (1984), and Kearsley (1985). It cannot be overstated that **the measurement instrument must match the educational objectives** – or else the entire evaluation process is of questionable value.

A third source of data, which may not be applicable to all programs, is that of frame analysis. If questions are interspersed throughout the program or decision points are created for the student (for example, which lab tests to order in the simulation of a doctor's gathering of patient data), it is useful to keep track of responses in order to detect areas within the program that have unusually high error rates. Those areas can then be examined to see if the problem is that the students lack the necessary background preparation for the question or if the question is poorly worded or designed.

When does formative evaluation end? Not until the instructional designers are convinced that the program is as free of "bugs" as possible and is producing consistently the results desired. Often the decision is an arbitrary one, since improvements can always be made, and is based on the perception that "investment in more formative evaluation is likely to result in diminishing returns" (Kandaswamy, 1980).

SUMMATIVE EVALUATION

Once formative evaluation is completed and the iterative process of obtaining feedback and making revisions is concluded, the instructional courseware is implemented with the actual target audience in order to judge its overall effectiveness, under the conditions in which it will actually be used. The purpose in conducting summative evaluation is often twofold: (1) to document and communicate the effectiveness of the instructional courseware to colleagues or other potential users, and (2) to be able to justify its use should a policy decision regarding its adoption or discontinuation need to be made.

One of the mistaken assumptions many developers have regarding summative evaluation is that there is only one criterion on which to judge the success of CBI: Do the students using CBI learn more than the students not using CBI? Such a reductionistic approach does not allow for exploration of many other critical factors to consider when trying to document the effectiveness of CBI. In addition to examining achievement, summative procedures should also include examination of attitudes, attrition rates, time spent by student to complete unit of instruciton, and cost to develop and support CBI. A brief discussion of each of these criteria follows.

ACHIEVEMENT: As in the case of formative evaluation, it must be stressed that whatever measures of achievement are used, they must match the instructional objectives of the courseware as well as the mode of interaction employed. Unlike formative evaluation, however, the results of these measures are often subjected to statistical analysis in order to determine the learning efficacy of the CBI. Common statistical experimental designs used include single–group pre–test/post–test design, nonequivalent control group design, and pre–test/post–test control group design (Muller, 1985).

Single–group pre–test/post–test design is often used when random assignment of students to treatment or control groups is not possible or when the creation of a control group is not feasible. In this design, the treatment is administered to the entire group of learners and pre–test and post–test measures obtained. Often, then, the means of the pre–tests and the post–tests are compared using a "t–test" for the statistical analysis. There are some serious limitations to the interpretation of the results of such analysis. Most obviously, it cannot be concluded that the actual change in scores was due to the treatment. As Muller notes, such factors as maturation, history, regression, instrumentation, selection, mortality, and interactions may confound the results and cause any conclusions drawn to be invalid.

Nonequivalent control group design is sometimes used as another alternative in the event that random assignment to control and treatment groups is not possible. In this design, the performance of the treatment group (i.e. the group using CBI) is compared with that of a "control" group – perhaps last semester's class or another section (not using CBI) of the same course. This design is often called a quasi–experimental design and because of its failure to meet the requirement of random assignment to groups, its results should not be analyzed via traditional parametric statistical tests. Unfortunately, traditional "t–tests" are frequently used nonetheless. It must be stressed that any results so obtained should be regarded with caution, if they are regarded at all! It is impossible in such a design to rule out confounding variables such as teacher and cohort effects, selection, equivalency of content covered, etc..

Pre–test/post–test control group design is, by far, the most statistically valid of the three experimental designs. Unfortunately, it is often the hardest to implement. In this design learners are randomly assigned to either a treatment group, which uses the CBI, and a control group which does not receive CBI but rather receives the material in the traditional method. Statistical tests such as the "t–test" or Analysis of Covariance (which can adjust for any pre–test differences between the two groups) are used to compare the differences of the group means. Threats to the internal validity of the results are the "John Henry effect" (i.e. the group or teacher not involved in the treatment determined to try to "beat the machine") or the "Hawthorne effect" (i.e. the students in the treatment group feel "special" and are motivated to outperform the other group). Equivalency of content covered must be insured in order to assume validity of results, i.e. any differences found must be able to be attributed to the method of instruction, not to a difference in material covered. There are many possible variations of this design, which is one of its advantages. For example, the control group could receive information in a lecture plus extra small group discussion while the treatment group could receive information in a lecture plus time spent with computer–based materials (for the same amount of time as the control group spends in discussion). For an example of a well designed study using this type of experimental design see Huntley, J.S., Albanese, M.A., Blackman, J.A., and Lough, L.K. (1985).

The above discussion of experimental designs certainly does not exhaust the various methods that can be used for determination of learning effectiveness. In addition, the yardstick of success is not an absolute one. In some cases it may be that the finding of "no differences" between treatment and control group is not of consequence (assuming that the treatment group performs as well as the control group). For example, if a program is designed to teach the students how to use a sophisticated piece of laboratory equipment (via a simulation) and allows for more students to be trained in the use of that instrument, then as long as the treatment group performs as well as, but not necessarily better than the control group on a measure of instrument use, its adoption as a teaching method can be justified and endorsed. Of course, nonsignificant differences due to large measurement variability represent an entirely different problem. On the other hand, if a program that is designed to teach students statistical concepts does no better than the traditional lecture and readings form the textbook method then one would have to seriously examine other factors, such as those that follow, to justify its adoption and use.

ATTITUDE: Procedures used to assess student attitudes toward CBI during summative evaluation do not differ from those used in formative evaluation. Emphasis is on creating a scale that will be specific enough to pinpoint exactly what the student's Generally, it has been found that students are very positive toward CBI (see, for example, Orlansky and String, 1981; Thomas, 1979;and Kulik, Kulik, and Cohen, 1980). However, one important comparative study involving the evaluation of PLATO and TICCIT in community colleges revealed interesting attitudinal differences (Alderman, Appel, & Murphy, 1978). Results showed that student reactions to PLATO were generally more favorable than those to TICCIT, although the PLATO system did not have significant impact on student achievement, whereas TICCIT did. The less positive feelings toward TICCIT seemed to be a function, in part, of class size and amount of individual attention received from the instructor. Thus, when judging attitudinal data it is worth the time and effort to identify those factors, besides the program itself, that contribute to the the students' affective outcomes.

ATTRITION RATES: A factor that can be judged during the summative phase ofevaluation, but not the formative, is that of attrition. How many students in the treatment group drop the course compared to those in the control group? Or is there a difference between the groups in the number of students who elect to go on to a next sequence level of the course? As is the case in statistical design, care msut be taken that differences are not due to confounding variables such as difference in student ability levels, teacher effects, etc.. If it can be demonstrated that a greater proportion of students complete a course as a result of CBI (as was the case with the use of Micro–Mate, a program used in medical terminology classes at Central Piedmont Community College, see Perkins and Squires, 1985), that can be a powerful argument for the continued use of the CBI program.

TIME SPENT BY STUDENT: Another factor to be examined in some uses of CBI (but not all – it obviously depends on the goals of the specific CBI involved) involves the amount of time it takes a student to master the learning objectives. Because the computer can offer individualized instruction, it is often a more efficient way to teach certain concepts, facts, or procedures. In a study by Orlansky and String (1981), it was found that CBI significantly decreased instruction time by an average of 30 percent. Likewise, studies reported by Thomas (1978), Abrahamson (1969), and Solomon (1974) all report that CBI consistently reduces the amount of time needed for instruction. However, in a study cited in Knapper (1981) regarding the evaluation of COMIT (Computerized Multi–Media Instructional Television project, a system of computer–assisted instruction developed by IBM), it was reported that students in the COMIT group spent an average of 8.7 hours at the computer terminals in comparison to the control group which spent an average of 6.1 hours attending lectures. Of course, variables that must be considered when studying time spent by students at the computer is the initial time it takes them to become familiar with the system and the use of the computer and whether they have learned the same amount as the students in the control group.

COST: The cost of CBI must be considered when examining its overall effectiveness. It is clear that CBI is not an inexpensive method of education either in terms of time taken to produce the courseware (and subsequent salaries for staff involved), hardware neede, or production costs (particularly in the case of interactive videodisk production, which can easily run into figures of $80,000 plus per hour of instruction). Selden and Schultz (1982) note, however, that "cost itself is a minor issue compared with cost effectiveness" (p.62). What the cost must be balanced against are such factors as the longevity of the courseware and hardware, maintenance costs, number of students who benefit from its use, its replacement of more expensive equipment (as is the case with spectrometer simulators in chemistry labs), reduced learning time, better use of instructor time, etc. Evaluation of cost, then, is not simply a matter of adding up the dollars and cents, but rather is based on many of the factors mentioned in the preceding pages.

Obviously, evaluation of CBI is not a simple process; it certainly is not something that can be accomplished at the "final hour", and administered much like a semester exam. Rather it is a process that must be planned for from the beginning. While computers have become regarded by some as the "Salk vaccine for the educational ills of the 80's" (Roblyer, p.40), what we know about their impact on how students learn needs to be further explored and documented. Well designed and implemented evaluation, both formative and summative, could be of great aid in this endeavor.

APPENDIX A

Necessary attributes to consider in designing courseware Cohen, 1983 in **Educational Technology**, 20 (1),p.10.

Generic to Instructional Design:

1. Target audience specified
2. Learner entry competencies specified
3. Rationale, goals, and objectives specified
4. Objectives stated behaviorally
5. Objectives stated in terms of the learner
6. Objectives include higher-order skills
7. Learners informed of the objectives
8. Range and scope of content adequate
9. Preinstructional strategies used:
    - Pretests
    - Advance organizers
    - Title at beginning of unit
10. Instructional text formatted for easy reading
11. Concept learning employed in the instructional approach
12. Vocabulary used appropriate for the learner
13. Graphics embedded in the context
14. Graphics used appropriately
15. Demonstration of the exercise provided
16. Teacher's manual provided
17. Instructions clearly stated for the student
18. Evaluation components provided

Necessary for Design of Courseware

1. Curriculum role used:
    - Adjunct
    - Mainline
    - Management only
    - Other
2. Mode of interaction employed:
    - Drill and Practice
    - Tutorial
    - Game

Simulation
Problem–solving

3. Student sequenced through the content:
    Nonlinear
    Varied by teacher/student
4. Instructional text formatted for screen display
5. Graphics embedded in the context
6. Graphics used appropriately
7. Cues and/or prompts used
8. Action occurs on the screen
9. User control granted to the learner
10. Computer–Managed Instruction employed
11. Feedback used appropriately
12. Records stored on magnetic devices for future retrieval
13. Content designed to be altered
14. Random generation used
15. Packaging designed for component parts
16. Teacher's Manual and Student's Manual provided
17. Technical design used:
    Quick response time
    Quick loading time

APPENDIX B

Sample verbal protocol from pharmacy project developed by Larry Boh Michael Pitterle, and Joseph Wiederholt, University of Wisconsin–Madison

I= Interviewer; S= Student

I: Why are you choosing to look at the lab tests?
S: To get some more information, this patient's diseases and things... the lab tests give more detailed information about the type of arthritis she has and things like that...All right, I'm going to look at hematology because there might be a sed rate [looks at the chart on the screen and giggles}...well, I don't know my normal values too well so this isn't going to help me as much as it should...
I: What are you thinking of now?
S: I'm trying to remember what things would be abnormal in rheumatoid arthritis and which lab tests I should look for to see those. But for rheumatoid factor. I'm trying to think how to get at that one. Well, I'll get back to that one. I'll look at the white blood cell differential to see if that shows elevated white cell count.
I: What does that tell you?
S: Not much because I need to know my normal values better.
I: What are you trying to decide now?
S: How to find rheumatoid factor. I suppose that might have been under hematology but I don't remember. I don't think it was though. I guess I'll look for special lab tests.

Later in the program..
S: Well, a lab test will have changed so I'll look at that. Uh, see if I can do better this time. I'm going to look at hematology.
I: What are you going to look for there?
S: Look for the sed rate. Okay, there's an elevated sed rate. Okay, I see. I didn't realize last time before that there's supposed to be another set of numbers down here. I thought that...so those are the normals– I didn't realize that. Ummm...okay.
I: What are you trying to decide?
S: Well, last time that it told me "no information" that was because there were no tests run but I thought it was a sign that I wasn't supposed to choose that so I'm wondering if I should go back to some of those that I looked at last time. I'll look at red cell morphology...I could be wrong again. there might be some

information there. Nope. And I still want to look for rheumatoid factor and I still don't know how to. ....

Later in the program...
I: Why are you going back to the primary patient data?
S: I think the case history is going to be the same...ummm...I don't know though, maybe I should see if that has changed...they may have added what...okay, "feeling better"...now, I'm not sure, I think the hydroxychloroquine has been started, but I really don't know...well, I guess I'll assume that it has, although it never asked about doses or anything like that, so I don't know but I think she's feeling better because of the hydroxychloroquine or maybe it's just because of the changed dosage in the prednisone and aspirin. I don't know.

APPENDIX C

Sample of questions used in an attitude survey designed by A.L. Schlack, Jr., Department of Engineering Mechanics, University of Wisconsin–Madison.

Directions: On a scale from 5 (highest) to 1 (lowest) encircle your assessment of the validity of the following statements (i.e. 5 means you strongly agree and 1 means you strongly disagree).

The instructions given for using the tutorial system are adequate.	5 4 3 2 1
The number of terminals and the hours they are available is adequate for a class this size, assuming sign up privileges are available.	5 4 3 2 1
I would rather go to a TA for help so I can see what is wrong with my own approach than to rely on a computer solution.	5 4 3 2 1
The computer solutions were very helpful to me with the following types of mistakes:	
(a) computational	5 4 3 2 1
(b) writing basic equations	5 4 3 2 1
(c) fundamental concepts	5 4 3 2 1
(d) free body designs	5 4 3 2 1

On a scale from 5 (highest) to 1(lowest) encircle your evaluation or rating of the following:

the importance of using colored monitors compared to black and white	5 4 3 2 1
the value of the sequential solution presentation, with the pace controlled by the student	5 4 3 2 1

APPENDIX D

Sample questions from student attitude survey by Larry Boh, Michael Pitterle, and Joseph Wiederholt, School of Pharmacy, University of Wisconsin–Madison

Directions: As a component of the computer simulation project, we would like you to identify your views about the use of computer simulations to facilitate the transfer of academic knowledge to an applied setting. Your views are important for the success of this project and the time you take to complete this survey is appreciated. Read each of the following statements and assign a number depending on the degree of agreement with the statement. If you are undecided try to pick up the response closest to your feelings the majority of the time. Honest and candid responses are extremely important.

Question #5
Using a computer in a course to do assignments worries me a lot.

1. Strongly agree
2. Agree
3. Disagree
4. Strongly disagree

Type a number between 1 and 4 for the best return and then <ret>

Question #14
I would like working with computer simulations as part of the therapeutics course.

1. Strongly agree
2. Agree
3. Disagree
4. Strongly disagree

Question # 19

The material presented in the computer simulation was challenging.

1. Strongly agree
2. Agree
3. Disagree
4. Strongly disagree

Question # 20

The purpose of the rheumatoid arthritis simulation was clearly defined.

1. Strongly agree
2. Agree
3. Disagree
4. Strongly disagree

Question #24

The amount of time to complete the simulation was too long.

1. Strongly agree
2. Agree
3. Disagree
4. Strongly disagree

Question #29

I do not feel the computer simulation tested my therapeutic knowledge about medications.

1. Strongly agree
2. Agree
3. Disagree
4. Strongly disagree

Question #35

I currently own or have access to a personal (micro–) computer.

1. Yes
2. No

APPENDIX E

Characteristics of various modes of interaction, examples (from Project TROCHOS holders at the University of Wisconsin–Madison) and measurement concerns

A. SOFTWARE THAT DEVELOPS KNOWLEDGE AND SKILLS

DRILL AND PRACTICE

Characteristics:

— focuses on factual knowledge and skill development of **previously** learned material
— goal is to have software focus on increasing students' speed at processing information
1. computer can record time it takes for student to carry out problem or recognize certain information
2. computer can adjust amount of time a problem is presented
3. criterion levels can be developed to assess whether students have achieved skills for automaticity
— can also have software focus on helping students bring together or chunk information
— used to supplement classroom teaching

Measurement concerns:

—Does student reach mastery or criterion level?
—Has student increased automaticity of skill or speed in retrieving information?

Example:

C.B. Thomas, S. Eicker., School of Veterinary Medicine.
Courseware for the Analysis of Herd Problems in Animal Production

We propose to develop courseware that will require and provide drills in the quantitative methods for describing disease events in animal populations, in analytical methods for identifying causal factors in the occurrence of disease, and in alternative methods of intervention to control disease with consideration of the consequent costs and benefits.

CONTENT TUTORIAL

Characteristics:

—differs from drill and practice by focusing on instructing students in new subject area facts and skills. Goal is to provide sufficient grounding in content and skill areas so that complex problems can be presented
—concept of chunking might be used to enhance effectiveness
1. use of mnemonic devices
2. semantic mapping
—computer needs to acquire information about the prior knowledge of students in order to relate new information to familiar concepts; makes use of branching procedures after original diagnosis
—could have diagnostic routines that identify the types of errors students make and then coach them in ways to avoid making errors
—also could test for misconceptions (see, for example, article by Hewson, 1984)
—intended to stand alone, not supplement classroom instruction
—includes feedback to student responses and branching programs which route students to different situations and exercises dependent on pre–established criteria

Measurement concerns:

—What misconceptions does the student have about the materials?
—Where is the feedback placed? What form does it take? What are the branching procedures?
—Possible measurement instruments include verbal protocols, similarity judgements, in addition to standard classroom tests

Example:

Leonard Uhr, Computer Sciences Department
Artificial Intelligence

We propose to develop a single basic set of learning/teaching tools, so that students in any specific artificial intelligence course, no matter what their prior background, will be able to ask for help, information, exercises to help learn, and programs to look at, modify, run, and rerun...A set of materials will be developed to help a neophyte: (1)learn to use a computer system, including the operating system (UNIX), programming languages (Pascal, Lisp), and editors (ex,vi), and (2) learn about computer vision, problem-solving, and other areas of artificial intelligence.

HEURISTIC TUTORIALS

Characteristics:

--designed to teach problem solving strategies; use a well-structured environment for training and practice in use of specific heuristics
--question of interest: Do strategies generalize?
--use of error diagnosis and coaching routines could be very beneficial

Measurement concerns:

--Can look at time to solution of problems, decision making paths
--Can compare performance of student to that of "expert" in field
--Verbal protocols, problem sorting, can be used for measurement
--need method of assessing degree of transfer to new problems, i.e. how well do the strategies transfer or generalize to other stimulus situations embodying other problems of the same type?

Example:

Michael Streibel and James Stewart, School of Education
Interactive Genetics Tutorial Project

The Interactive Genetics Tutorial project will help students gain a high level of expertise and understanding in transmission genetics and problem-solving skills. To accomplish this, we will have to do the following: (1) program a genetics problem-generator on an IBM PC AT in a language such as Modula II, (2) collect existing still and moving visuals about plant, insect, and human genetics and transfer them to a video environment, (3) integrate the genetics problem generator and video archive into an interactive video system, and (4) add an expert tutor component to the above system.

The various components of the Interactive Genetics Tutorial are instructionally innovative for several reasons. The genetics problem generator permits an unlimited number of inheritance patterns to be created within instructor-controlled parameters. It is therefore truly a genetics "problem–solving kit." This flexibility and range approximates real–life genetics problem solving. The videodisk component permits visual pectures to be used as models and explanations and therefore opens up a whole new set of real–life problems for the students to solve.

B. PROBLEM SOLVING SIMULATIONS

–– pose complex problems for solution
––present situations where problem needs to be identified, where there may be large variety of possible heuristics to use and a number of acceptable solutions

CONTENT / INSTRUMENT SIMULATION

Characteristics:

––designed to provide environment as lifelike as possible for student to use in problem–solving activity or in acquiring skills in operating an instrument
––modifications that may be useful
1. build tutorial software into system via menu (such as technical terms needed)
2. build in error detection and coaching routines
––stresses active processing of information and procedural knowledge
––assumes that basic concepts and principles have already been learned

Measurement concerns:

––How well does knowledge acquired in simulation transfer to real life situations or real instruments?
––What is the relationship between the knowledge of course content and the application of correct problem solving procedures?

Example:

Phillip Certain and Paul Schatz, Dept. of Chemistry
Two Projects in Chemistry Education

The primary focus of the organic chemistry laboratory is synthesis and characterization. After a compound is synthesized, its identity is confirmed by one or more analytical techniques, often by a spectroscopic method in which the spectrum of the compound is obtained and compared with spectra of known compounds or with theoretical predictions.

The computer has already been introduced into these laboratories in a very innovative way by Dr. Paul Schatz. With a program written for the IBM PC, he can now simulate the console of an NMR spectrometer, probably the most ubiquitous analytical instrument available to the organic chemist.

ARTIFICIAL ENVIRONMENT SIMULATIONS

Characteristics:

--generate a make-believe world with a sufficiently **rich knowledge base** and set of rules that are **complex**

Measurement concerns: same as those above

Example:

Gregory Moses and Michael Corradini, College of Engineering
Computing Equipment for the Development of a Nuclear Fission Reactor Simulation Laboratory

A nuclear power plant simulator must model the reactor kinetics, plant thermal hydraulics and plant control systems to provide an effective simulation. Models of all these components are well developed at various levels of sophistication. We propose to develop courseware to implement the appropriate models as a set of first order coupled differential equations and algebraic equations. These equations will be numerically solved in a program written in Fortran 77. This numerical program will interface with graphical displays on the screen in the form of dials, strip charts, actuator lights, and other forms of display common to nuclear power plant control rooms. The screen would include a clock to identify the simulation time. Another display option would be to show a schematic of the power plant. In this schematic, color graphics would characterize such things as the temperature and flow of the coolant. Such a simplifies simulator would allow engineering students to test their understanding of the dynamics of nuclear power plants under different transient scenarios.

C. PROBLEM SOLVING AIDS

TOOLS

Characteristics:

--contain programs that require the student to actively use them in ways that will help solve problems which may be either self-defined or externally designed e.g. data management systems, word processing information retrieval systems

Measurement concern:

--How well does the student master the procedures needed to effectively use the programs?

Example:

Robin Chapman, Dept. of Communicative Disorders
Instructional Computing for Instruction in Communicative Disorders

... A second primary consideration is that the field of communicative disorders has approached a crossroads in clinical information management and decision making. Many state-of-the-art clinical procedures require the collection and analysis of extremely large amounts of data.

Educational programs in communicative disorders must lead the effort in (1) developing computer-assisted clinical assessment and management, and (2) preparing graduates who can effectively use this technology. The advantages of this technology are considerable to professionals who must manage large amounts of information and often derive summaries and patterns from complex speech and language samples.

PROGRAMMING LANGUAGES

Characteristics:

--vary according to language learned

Measurement concerns:

--What is the effect of prior knowledge of a different programming language?
--Does the student have sufficient mastery of the programming language to meet his/her needs?

Example:

Edward R. Pope, School of Education
Microcomputer Assistance in the Instruction of Art and Design

...From my experiences over the past three years in gathering resources for teaching and teaching both beginning and advanced computer–mediated art courses, I see three great needs. A third necessity is software for teaching programming languages that have a graphic orientation and for teaching graphic programming that has an art and design orientation. Generally, the examples found in language instruction seem to have a business, math, or scientific application. Specifically, graphic programming instruction presently seems intended for computer programmers or visually naive audiences. This imposes a feeling of irrelevancy for art and design students.

D. REAL TIME USE

Characteristics:

-- interfacing computers with experimental apparatus in order to bring experiments which are normally unavailable into student lab.

Measurement concerns:

--Is the interfacing cost effective?
--How much and what does the additional experience add to the student's learning in the content area?

Example:

Paul Milenkovic, College of Engineering
Workstation for Digital Signal Processing Analysis of Audio Signals

The project goal is no less than to develop the counterpart to the word processor and the spreadsheet for the digital analysis of signals in the audio frequency range. The capabilities of this workstation for speech analysis are currently available only at a few highly funded research lab facilities in the nation. Making these capabilities available on a widespread basis in college instruction would have a revolutionary effect on the teaching of speech technology as well as other areas involving the analysis of audio band signals.

REFERENCES

Abrahamson,S., Wolf, R.M., & Denson,J.J. (1969). A computer based patient simulator for training anesthesiology. **Educational Technology**, 9 (10), 55–59.

Alderman,D.L., Appel,L.R., & Murphy, R.T. (1978). PLATO and TICCIT: An evaluation of CAI in the community college. **Educational Technology**, 18 (4), 40–46.

Bell, M.E. (1985). The role of instructional theories in the evaluation of microcomputer courseware. **Educational Technology**, 25 (3), 36–40.

Billings, D.M. (1985). An instructional design approach to developing CAI courseware. **Computers in Nursing**, 3 (5), 217–223.

Brown, B.R., & Ellinger, R.S. (1981). Evaluation of computer–based instructional material: Intent and procedures. **Topics in Educational Computing**, 22–33.

Clark, R.E. (1982). Antagonism between achievement and enjoyment ATI studies. **Educational Psychologist**, 17 (2), 92–101.

Cohen, V.B. (1983). Criteria for the evaluation of microcomputer courseware. **Educational Technology**, 23 (1), 9–14.

Cohen, V.B. (1985). A reexamination of feedback in computer–based instruction: Implications for instructional design. **Educational Technology**, 25 (1), 33–37.

Dick, W. , & Carey, L. (1985). **The Systematic Design of Instruction.** Glenview, Illinois: Scott, Foresman, and Company.

Duquette,C. (1985). Formative evaluation: One instrument. **Educational Technology**, 25 (2), 20–23.

Ericsson, K.A., & Simon, H.A. Verbal reports as data. **Psychological Review**, 87 (3), 215– 251.

Fetter, W. (1984). Guidelines for evaluation of computer software (with an evaluation form). **Educational Technology**, 24 (3),19–21.

Forman, D. (1982). Courseware evaluation for curriculum integration. **Monitor**, April/May/June, 7–9.

Golas, K.C. (1983). The formative evaluation of computer-assisted instruction. **Educational Technology**, 23 (1), 26-28.

Grabowski, B. and Aggen, W. (1984). Computers for interactive learning. **Instructional Innovator**, 29 (2), 27-30.

Hewson, P.W. (1984). Microcomputers, conceptual change, and the design of science instruction: Examples from kinematics and dynamics. **South African Journal of Science**, 80, 15-20.

Huntley, J.S., Albanese, M.A. Blackman, J.A., & Lough, L.K. (1985). **Evaluation of a computer-controlled videodisc program to teach pediatric neuromotor assessment.** Paper presented at the annual meeting of the American Educational Research Association, Chicago.

Jay, T.B. (1983). The cognitive approach to computer courseware design and evaluation. **Educational Technology**, 23 (1), 22-25.

Jerstedt, C.G. (1982). Active learning increases educational effectiveness and efficiency. **T.H.E. Journal**, May, 97-105.

Kandaswamy, S. (1980). Evaluation of instructional materials: A synthesis of models and methods. **Educational Technology**, 20 (6), 19-25.

Kearsley, G. (1984). **Computer-based training: A guide to selection and implementation.** Reading, Massachussetts: Addison-Wesley Publishing.

Kearsley, G. (1985). Helps, simulations, and intelligent tutors: An overview of embedded training. **Data Training**, May, 32-38.

Kearsley, G., & Hillelsohn, M.J. (1982). Human factors consideration for computer-based training. **Journal of Computer Based Instruction.**

Kearsley, G., Hunter, B., & Seidel, R.J. (1983). Two decades of computer based instruction projects: What have we learned? **T.H.E. Journal**, February, 88-96.

Knapper, Christopher.(1980). **Evaluating instructional technology**. New York: Halsted Press.

Kulik, J.A., Kulik, C.C., & Cohen, P.A. (1980). Effectiveness of computer-based college teaching: A meta-analysis of findings. **Review of Educational Research**, 50 (4), 525-544.

Lai, M.K. (1984). **Evaluating a computer education program qualitatively and quantitatively.** Paper presented at the annual meeting of the American Educational Research Association, New Orleans.

Larkin, J., McDermott,J., Simon, D., & Simon, H. (1980). Expert and novice performance in solving physics problems. **Science**, 208, 1335-1342.

Lehrer, R. (1986). **Logo as a Strategy for Developing Thinking?** il/May/June, 5-6. Educational Psychologist,in press .

Loyd, B. H., & Gressard, C. (1984). The effects of age, sex, and computer experience on computer attitudes. **AEDS Journal**, Winter, 67-73.

Mackin, E., & Suppes, P. (1976). Evaluation studies on CCC elementary school curriculum 1971-75. **CCC Educational Studies**,1 (1).

Muller, E.W. (1985). Application of experimental and quasi-experimental research designs to educational software evaluation. **Educational Technology**, 25 (10), 27-31.

Orlansky, J., & String, J. (1981). Computer-based instruction for military training. **Defense Management Journal**, 2nd quarter, 46-54.

Patterson, J.H., & Smith, M.S. (1985). **The role of computers in higher order thinking.** University of Wisconsin: Wisconsin Center for Education Research.

Perkins, B., & Squires, C. (1985). Tutoring with microcomputers: The micro-mate project. **Educational Technology**, 25 (4),42-43.

Roblyer, M.D. (1985). The greening of educational computing: A proposal for a more research-based approach to computers in instruction. **Educational Technology**, 25 (1), 40-44.

Rothkopf, E.Z. (1963). Some observations on predicting instrumental effectiveness by simple inspection. **Journal of Programmed Instruction**, 2, 19-20.

Salomon, G., & Gardner, H. (1986). The computer as educator: Lessons from television research. **Educational Researcher**, 15(1), 13–19.

Schlack, A. L. Jr. (1985). **Computer aided tutorial system for statics.** Unpublished paper. University of Wisconsin–Madison.

Selden, P.H., & Schultz, N.L. (1982). What the research says about CAI's potential. **Training/HRD,** November, 61–64.

Solomon, L. (1974). CAI: A study of efficiency and effectiveness. **Educational Technology**, 14 (10), 39–41.

Steinberg, E. (1983). Reviewing the instructional effectiveness of computer courseware. **Educational Technology,** 23 (1), 17–19.

Thomas, D.B. (1978). The effectiveness of computer–assisted instruction in secondary schools. **AEDS, 12** (3), 103–116.

Wade, T.E. Jr. (1980). Evaluating computer instructional programs and other teaching units. **Educational Technology**, 20 (11), 32–35.

Wager, W. (1982). The software evaluation dilemma. **Monitor,** Apr/May/June, 5–6.

INDEX BY PRESENTER / AUTHOR, VOLUME IV

Presenter / Author **Page**

Alexander, Kirk 152

Billman, E. 38

Bogom, Stuart 65

Bowes, John E. 2

Braca, Michael J. 92

Braune, R. J. 38

Churchill, Lynn 111

Comeaux, Michelle 170

Danielson, Wayne 102

Davis, T. 38

Entwistle, W. C. 38

Flach, J. M. 38

Gillman, Clifford 170

Golde, Hellmut 18

Guenin, Mark A. 11

Helm, Patricia Burton 54

Horowitz, Ellis 98

Lintern, G. 38

Mahoney, M. 60

Mane, A. 38

Mintz, Norman 23

INDEX BY PRESENTER / AUTHOR, VOLUME IV

Presenter / Author **Page**

Oppenheimer, Joe 83

Robert, Jean–Christophe 156

Schmitt, Gerhard 156

Sherwood, Bruce Arne 29

Sherwood, Judith N. 29

Siler, Randy 111

Stenstrom, Michael K. 76

Stokes, A. F. 38

Tettleman, Bruce 73

van Dam, Andries 92

Weller, M. H. 38

Wickens, C. D. 38

Winer, Mark 83

INDEX BY UNIVERSITY, VOLUME IV

University **Page**

Brown University 92

Carnegie–Mellon University 29
156

Columbia University 23
73

Princeton University 60
152

University of California, Los Angeles 76

University of Illinois 38

University of Maryland 83

University of Pennslyvania 54

University of Pennsylvania 11
65

University of Southern California 98

University of Texas at Austin 102

University of Washington 2
18
111

University of Wisconsin, Madison 170

IBM

Academic Information
Systems Proceedings

1986 University AEP Conference

Volume V:

ENGINEERING

April 5 - 8, 1986

Sheraton Harbor Island Hotel

San Diego, California

CONTENTS, VOLUME V

Title **Page**

A Microcomputer Tool to Design Energy Conserving Buildings ....

A PC-Based VLSI Design System for Education 16

An Instructional Facility for Computer Vision/Image Processing 25

An Interactive System Approach to Near Optimal Design 36

Closed-Loop Dynamic Testing Control Using an IBM PC/AT 57

Computer-Aided Design of Microwave Integrated Circuits 73

Computer Graphics Flow Visualization 79

Computer Aided System Analysis and Design 91

Computerized Instruction in Engineering Thermodynamics 95

Development of an Expert System for CAI in Process Control 104

Educational Software for Structural Engineering ....

Interactive Graphics in CAD/CAI in Chemical Engineering ....

Interactive Microcomputer Graphics Software/Modeling Phenomena 168

MCADA: Mechanical Computer Aided Dynamic Analysis 179

Teaching Numerical Math Using Interactive Computer Graphics 191

The Use of Interactive Thermal Finite Elements 196

VEGA: A Geometric Modelling System ....

Which Wave Theory 215

INDEX by Presenter / Author, Volume V 231

INDEX by University, Volume V 233

A PC-BASED VLSI DESIGN SYSTEM FOR EDUCATION

Presenter: Mani Soma *University of Washington*

ABSTRACT

A VLSI design system has been installed on IBM PC-ATs for teaching graduate and undergraduate courses in integrated circuit design at the University of Washington. Traditional design systems, such as main-frame computers or workstations, are both costly and unable to handle large numbers of students in these courses. The PC-based system, offering a reasonably complete set of tools for layout, has been successfully installed (as of April 1986). Since each PC is a single-user system, the performance of these tools exceeds that of larger time-sharing computers assuming more than 4 users are doing layout at the same time. This paper presents the computing strategy and specific technical implementation of the VLSI tool set on the PC-ATs, reviews the successes as well as problems in our efforts, and discusses future plan for tool improvements.

1. INTRODUCTION AND PROBLEM STATEMENT

The revolution in Very Large Scale Integration (VLSI) started in university environment, and ever since the pioneer work by Mead and Conway [1], universities have been quite active in developing software tools for integrated circuit design. Several successful efforts at Berkeley, North Carolina, and IT among others, have led to design systems based usually on VAX computers (made by Digital Equipment Corporation) and on various graphic display stations (e.g. AED, Metheus, etc). While these systems have been widely distributed and used in VLSI education, they share two major drawbacks :

[1] The hardware cost is rather high for university budgets. A minimal system with a microVAX and a display station costs at least fifty thousand dollars ($50,000), not counting the recurring annual maintenance and support price. To support a reasonable number of students, say 30 per course, several display stations and a larger VAX are needed, making the cost more prohibitive, especially for universities in the development phase of their VLSI design program.

[2] The response of the design system in the interactive mode is rather slow. With more than 4 users doing layout on a VAX 11/780, the time taken to execute a layout command becomes too long to be acceptable. The situation worsens if there are background simulation or design rule checking jobs, with the result that most user's time is wasted in waiting for system response.

These two drawbacks were evident in our design courses offered during the academic years 1983 – 1985. Enrollment had to be limited to 15 per course and system resources quickly dried up as students attempted to do more complex designs. Any further expansion of our VLSI design program would call for either a large budget or a more creative strategy to match our computing resources to educationa lgoals.

This paper presents one such strategy, whose implementation over the past year has reached the first milestone : providing improved access and productive use of layout tools to a larger number of students. In the following sections, the paper will discuss the computing strategy, specific implementation issues, and some early results based on students' use of the system in the design courses in Winter quarter1986.

2. COMPUTING ENVIRONMENT AND STRATEGY

The overall computing environment, shown in Figure 1, includes :

[1] A microVAX–II with one Metheus display station and several alphanumeric terminals. This is the minimal system used initially for the VLSI courses.

[2] More than one dozen IBM PC–ATs acquired by the Olympus program. These PCs were actively used in other courses but not in VLSI due to lack of software.

[3] BITNET linking the PCs and the microVAX. This network is actually University–wide, even though it is shown in within the context of this paper.

[4] Common peripherals : plotters, printers, etc.

It is immediately obvious from the resources available that the PC–ATs should be put to better use in VLSI instruction. The key problem is software compatibility. VLSI software usually resides on a VAX and operates under the UNIX operating system. The tools range all the way from circuit simulators (e.g. SPICE from Berkeley) to logic simulators (e.g.RNL from MIT) to layout system (e.g. the Caesar system from Berkeley). It is clearly impossible to port all the tools to the PC-AT due to limited disk space and lack of an operating system compatible with UNIX. However, the bottleneck in our design courses does not involve the access to the entire software system; rather it is the access to the layout tools since these tools are most computing intensive as well as require the most expensive equipment. The decision was thus made to port the layout tools, or portion thereof, to the PC-AT as an experimental project to see whether the problem of access versus cost could be solved within a reasonable time frame. The goal is modest at the beginning : the PC-AT software should permit students to do as much of the layout task as possible without having to waste time to wait for the system response, which was the crucial problem with the microVAX. In fact if the user's waiting time could be cut byhalf, that would be a definite improvement. Further consideration of trade-offs between the two systems, VAX and PC-ATs, led to the following implementation strategy :

[1] The entire layout tool system, which includes layout definition, design rule checker, and electrical rule checker, is too large for the PC-AT disk (20 Megabytes) since the user will also need disk space to store layout files. The design rule checker and the electrical rule checker, which are quite computing intensive for circuits of reasonable size, are usually run in the background and thus do not have to reside on the PC-AT.As long as the layout files can be sent to the VAX for these verifications, the user needs not be concerned with running them interactively.

[2] The layout of the entire circuit does not have to be carried out on the PC-AT, especially if the circuit has a large number of transistors (50,000 or higher). The layout files of these circuits need disk space, and the PC-AT color graphic display lacks the physical size and resolution to show these circuits with sufficient details for visual inspection and checking. In fact the placement and wiring of cells to create the final chip layout usually cannot be done on the PC-AT even if the cells are displayed as rectangles with no internal details. These factors restrict the layout tools on the PC-AT to the creation of small circuits and the assembly of these circuits or cells into larger cells up to a certain limit before a larger monitor is required to continue the task.

[3] The software rewrite task appeared to be a herculean effort upon a closer inspection of the Berkeley Caesar programs. The lack of the UNIX operating system on the PC–AT further complicates this task. The graphic display driver also poses problems at the lowest level due to the hardware incompatibility between the PC–AT display and the Metheus (or AED) display. These considerations led to the decision that the layout files created on the PC–AT should be compatible with the VAX layout files and tools, without requiring that the layout tools must be identical. This decision permitted more implementation flexibility at the PC level without affecting the overall use of our design system.

Within the context of these tradeoffs, the project calls for the implementation of the layout creation tools on the PC–AT which permit the physical designs of circuits up to some size (to be determined during the implementation phase), respond quickly to user's commands in the interactive mode, and are compatible with the VAX layout tool system. The detailed implementation is summarized in the nextsection.

3. SPECIFIC IMPLEMENTATION

The decision not to rewrite the Caesar tools for the PC–AT does not imply our intent to reinvent the wheel by developing a totally new layout system. The cooperation between the Boeing company and the University of Washington in the early days of VLSI (1975) resulted in a PASCAL–based layout program called PLAP (Pascal Layout Artwork Program). PLAP takes an input file, which describes a circuit in terms of rectangles and layers, and compiles it to produce the graphic layout. PLAP was used for several years then dropped since it is inefficient compared to the Berkeley tools. However, with some improvements as described below, PLAP turns out to be quite easy to be implemented on the PC–AT. The changes in the PASCAL codes are minimal, and the major tasks are the graphic device driver, the cursor interface, and the mouse interface to eliminate the need to "write" a Pascal program to define a layout. The Professional Graphic Display was chosen, mainly because it was available along with the PC–ATs from the Olympus program.

The total effort of re–writing some PASCAL routines for the PC–AT, the graphic device driver, and the cursor interface took one man–year. We also developed a menu–driven input capability to simplify the use of the layout tool and with the cursor interface, eliminated the requirement of an input file. The output file from PLAP, in the earlier implementation, is in a database format incompatible with any other standard format for layout description such as CIF or GDS–II. Several new tools are written to translate the PLAP out–put format to these standard formats, which satisfy the compatibility requirements between the PC–AT and the VAX layout tool systems. We did not implement the mouse in the first pass since it is not as crucial as the verification that the layout tools work on the PC–AT and meet the goals mentioned above.

Several new capabilities are added to PLAP, especially in the interactive display of the circuit layout. Options are available for zoom, pan, moving around the layout, hierarchical display, display of selected mask layers, selection of details to be shown, store a specific display screen for later inspection and quick retrieval, etc.

The software configuration, after the first milestone of the PC–AT tool implementation, is shown in Figure 2. The configuration is limited to describing how the VAX and PC systems interact from the layout standpoint. Other issues concerning the simulation and test tools are not germane to this work, and are eliminated from this figure.

4. SYSTEM EVALUATION

As usual in this type of development project, we claim all the successes as well as the tribulations during the year–long effort. The successes are many, but unfortunately so are the tribulations and errors.

The first success is the solution of the access versus cost problem. The PC–ATs are low–cost graphic display stations compared to the others mentioned above, especially since our PC–ATs were acquired by an equipment grant. Instead of having 15 students fighting for access to the one Metheus display, we now leisurely accommodate between 35 to 50 students on five PC–ATs with Professional Graphic Display monitor. The second success is the response speed of the PC–AT system: the AT is extremely fast in responding to user's menu command and refreshing the screen. Due to the lack of the mouse, the cursor movement is not as fast, and is the only major deficiency in the system speed. This deficiency is being corrected at this time. The third success is actually a surprise even to the developers : the PC–AT can be used to layout circuits with up to 50,000 devices, much larger than we expected. In fact some students' projects can be laid out entirely on the PC–AT without resorting to the VAX for the final chip assembly. The fourth success is the compatibility between the two design systems: we have had no problems in sending layout files from the PC-AT to the VAX for design rule check and electrical rule check. The interface might be called cumbersome by the presence of the data-base translators, but at least it facilitates system communication at this time.

The major problems encountered in this effort all have to do with software and hardware documentation. PLAP itself is not well documented, which makes the system-dependent codes difficult to rewrite. The Technical Manual for the Professional Graphic Display does not contain sufficient information to enable faster development of the device driver. In fact, we discovered several errors in the Manual and finally had to resort to trial-and-error technique to find out how to communicate with the display. We have been hesitant in the mouse interface due to the concern that too much time might be wasted just in figuring out the documentation.

To further complicate these development problems, we contributed quite a few errors on our own, most of which have been corrected but several are still lurking around to surprise unsuspected users. The cursor movement, for example, sometimes refuses to move to the left but should be corrected soon since we just found out where the "bug" actually is at the time of this writing (March 1986). We complained about the poor source code documentation in PLAP, but the PC-AT version is only slightly better. The User's Guide to the PC-AT design system is more or less a set of notes collated and stapled together, making it difficult for new users to learn the system without a hand-on session taught by the software developers. There are of course more errors and problems waiting to be discovered, and we pause at this point to summarize the development effort and consider future improvement plan.

5. CONCLUSION

The first milestone of putting layout tools on the PC-AT is completed, and the successes are quite encouraging. Most of the project goals have been met, and despite some problems with the system at this time, we are prepared to add on more tools to make the PC-ATs the main design stations for VLSI instruction. These tools include simulation capabilities, test generations, and interface to test equipment. Parallel with these efforts is a clean-up project, which seeks to improve system documentation and users' interface to make the system more friendly and easy to use.

While we have solved the access versus cost problem, we do not claim that the PC–AT design system is "bett" than the VAX–based system. As shown in our system configuration (Figures 1 and 2), the PC–AT is only a part of the overall environment, and some mainframe computer (VAX or any othercomputer) is still needed for the computing intensive tasks mentioned in the paper. The larger and more expensive graphic displays sometimes are still needed for large layouts exceeding the PC–AT capabilities. The emphasis of our work is not to compare systems but to match the computing resources to the educational goals, and the development efforts presented herein should be viewed from this perspective.

6. ACKNOWLEDGEMENTS

Silicon Design, Inc. (Bellevue, Washington) provides extensive assistance in this software conversion and implementa–tion project.

REFERENCES

1. C.A. Mead and L. Conway, "Introduction to VLSI Systems", Addison–Wesley, 1980.

COMPUTING ENVIRONMENT

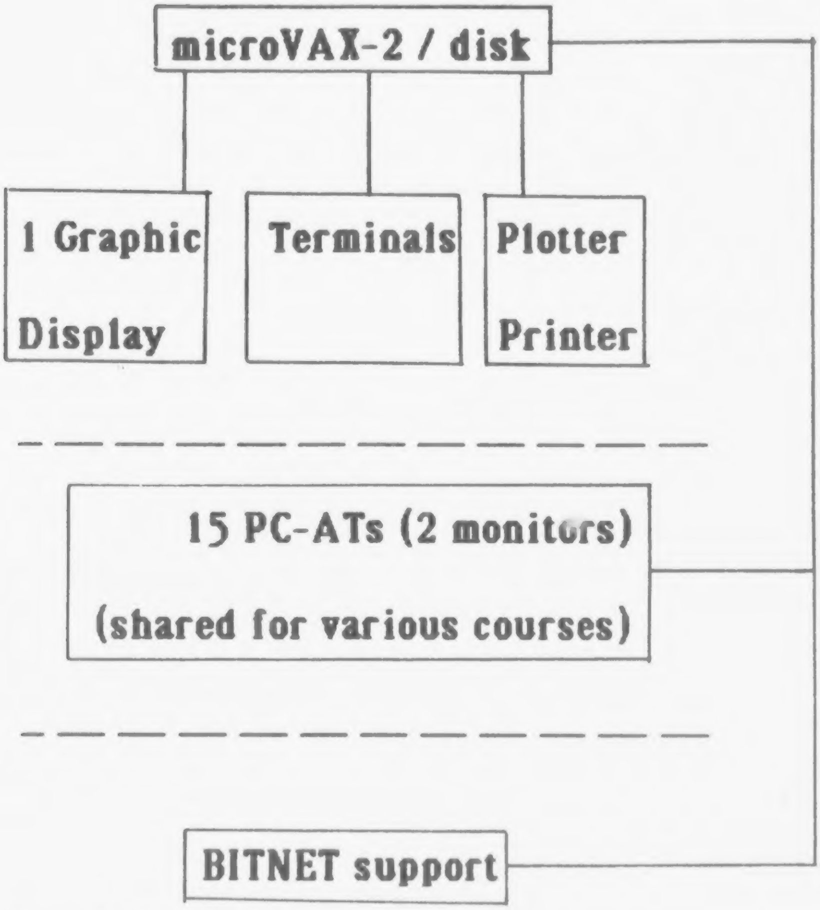


Figure 1. Computing Environment.

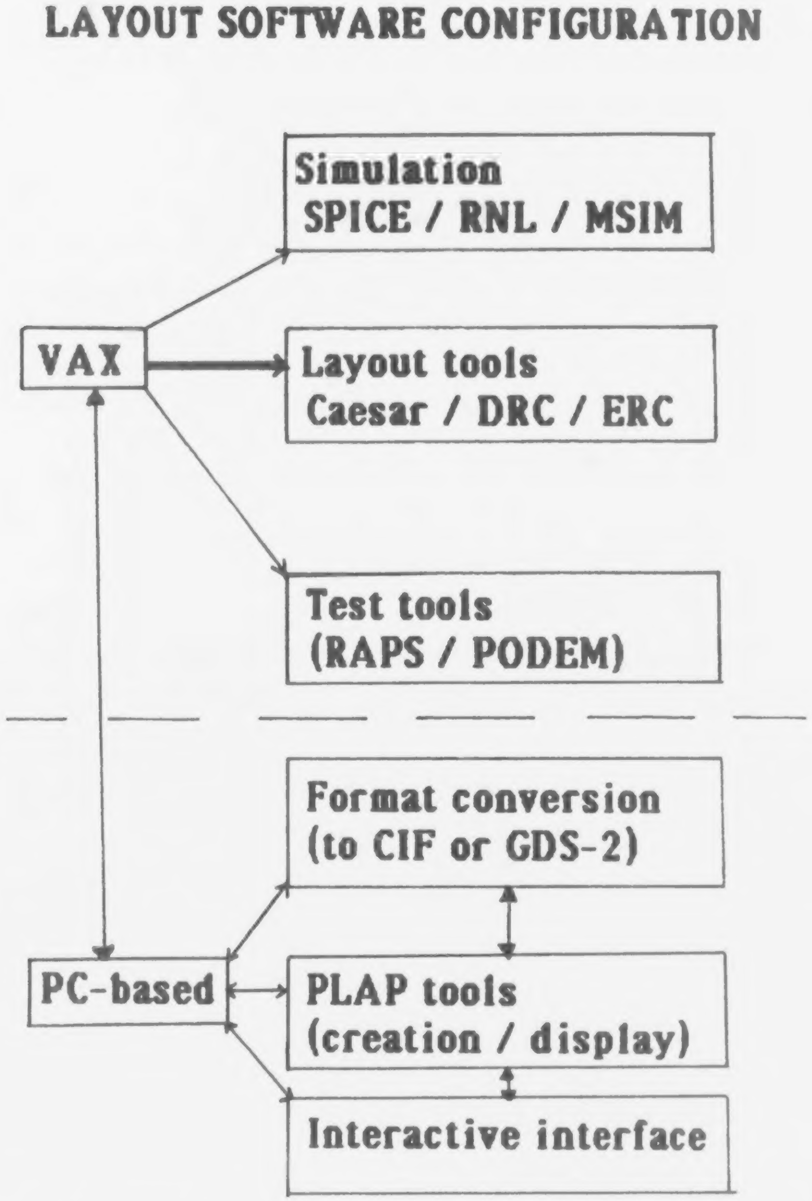


Figure 2. Layout Software Configuration.

AN INSTRUCTIONAL FACILITY FOR COMPUTER VISION AND IMAGE PROCESSING

Presenter: Anthony P. Reeves *Cornell University*

ABSTRACT

Computer vision is an emerging discipline in which most techniques are still very experimental in nature. An important part of computer vision and image processing education is direct experience in manipulating real image data. A computer facility to conduct experiments in computer vision and image processing must have the capability of storing and processing large image matrices (typically 256000 elements) and must have an image display device. An instructional environment for computer vision and image processing has been established on an IBM PC/AT system running the XENIX operating system. The professional graphics display is used as an image display device. A software system for computer vision, consisting of over 150 program modules, has been ported from a UNIX research environment tot he PC/AT.

INTRODUCTION

Image processing is concerned with the manipulation of the video data, such as digitized TV image or a photograph; an image is represented in the computer as a very large matrix of brightness values and typical image processing operations such as noise filtering and geometric warping require a large amount of computation. The results of an image processing operation are usually best perceived by a human when presented on an image display device such as a video monitor. Computer vision is concerned with the extraction of feature information, such as the location and identification of objects, from an image. Initial image preprocessing is usually essential to achieve good results from a computer vision system.

An instructional facility for image processing and computer vision has been developed which provides a wide variety of tools for the manipulation and display of image data. The goal of this facility is to provide students with experience in manipulating video imagery so that they better understand the behavior on real image data of image processing and computer vision algorithms. Students are expected to first use the system as a set of canned programs and then to develop an algorithm (program) of their own. At all stages the display of intermediate image results is very important; in some cases, interactive manipulation of the display is also necessary. At the end of the course the students make class presentations of their projects using slides photographed from the display device of the images they have computed.

The system is hosted on an IBM PC/AT running the XENIX operating system and currently, the IBM professional graphics display is used for image display. The tools consist of over 150 program modules [1] which have been ported from a UNIX based research system [2,3]. New tools are being developed to simplify the system use. Students' programs can be developed in C, Pascal, Fortran or even C Shell, whichever is most appropriate; program development tools are provided for all cases.

The main problems encountered in porting the system were attributed to deficiencies in the Microsoft compilers. The C compiler was provided with the system; the Pascal and Fortran compilers were obtained directly from Microsoft. For both the C and Pascal the 64k byte segment restrictions caused the most problems. The Fortran compiler is able to allocate large arrays (greater than one segment) but it has a number of other problems.

A major challenge of this project was to make the facilities of the research system available at a reasonable cost so that a number of systems could be used in an instructional environment. The PC/AT just met the minimum acceptable requirements for an image processing system in terms of memory and display capabilities. Last semester, this system on 4 PC/ATs was successfully used in a computer vision course which had an enrollment of over 40 students. All students gained initial experience on these PC/ATs, then they were used for 15 program development projects most of which involved groups of two students. Nearly all the projects were successfully completed. This semester a set of 9 PC/AT systems is being used to support a number of image processing courses.

THE HARDWARE FACILITIES

The host system from which the software system was ported is a typical research environment for computer vision and image processing research. It consists of a VAX 11/780 computer running UNIX 4.2 BSD. The image display device is a Gould DeAnza FD500. The host is a virtual memory machine with over 1 Gb of disk storage. The image display system frame memory is a 512 x 512 matrix of 24-bit pixels for true color images 8-bits are assigned to each of the three primary colors. The display also has hardware pan, zoom look-up-table and cursor functions. The display memory can be considered as three 8–bit deep images which can be individually displayed either as a monochrome image with 256 grey levels or with pseudo color mapping. The look–up–table (LUT) facility associates a 256 word 8–bit wide memory with each 8–bit color displayed. Instead of directly displaying a pixel, it is first used as an address into this memory and the contents of the selected word is the value which is actually displayed. In this way the user may rapidly define any point transform between the image and the display. In the pseudo color mode all LUT's are connected to a single 8–bit image; therefore, a unique 24–bit color pixel value may be assigned to each 8–bit pixel value.

Higher resolution, more expensive image display systems are used in some research facilities. A high resolution image display is typically 1024 x 1024 by 24–bit pixels. The FD500 has an enhanced pseudo color mode which displays 12 bit pixels via three 4096–word LUTs.

The educational system consists of an IBM PC/AT with over 1Mb of main memory and a 20 Mb hard disk. The display device is currently the IBM professional graphics display. This device is oriented towards computer graphics applications in that it has special hardware for drawing vectors but it only has 8–bits of information per pixel. The display size is 640 x 480 matrix of 8–bit pixels. A further complication is that when using the display for monochrome imagery only 16 grey shades can be displayed since the hardware only permits 4 bits of information to be associated with each primary color. There is a LUT capability which maps the 8–bit pixels into 12–bits (4 for each color). There is no zoom pan or cursor facility; furthermore, the interface to the host system does not have a very high bandwidth. It takes about half a minute to write a 512 x 512 image to the display memory from the host. A number of techniques which were developed to overcome the limitations of the display system are discussed in the next section. The system also had a monochrome display, which was well worth the additional cost, and a graphics printer.

SPECIAL TECHNIQUES FOR IMAGE DISPLAY

The software is set up to display 512 x 512 images; however, the 640 x 480 aspect ratio of the professional graphics was not a major problem. The central 512 x 480 region of the screen is used for image data display; the lowest 32 lines of the image data are not displayed. The left and right boarders of the display screen are reserved for other information; currently a grey scale is usually displayed in the left boarder region. A feature of the software system is to divide the display into 4 256 x 256 quadrants so that images may be conveniently compared. This was very useful for the IBM system since the method of multiple image frames cannot be used. The lower two quadrants are 256 x 224 in size.

FULL 8-BIT GREY SCALE

Most of the imagery to be processed is a monochrome with 8-bit pixels. The display restriction of only 16 viewable grey shades was a serious problem. A small external hardware adaptor (consisting of four resistors and a switch) was developed to display the full 256 grey shades. This adaptor combines 1/16 of the blue signal with the sum of the red and green signals; the result is connected to all three color inputs of the monitor. The full 256 grey scale is viewable when the look-up-table is set up so that the four most significant bits of each pixel is mapped to red and green and the least significant four bits are mapped to blue.

DITHERING

An alternative technique to provide improved grey scale display without additional hardware is dithering. This technique is suitable when an image is enlarged for display. For example, much work is done with 128 x 128 test images, these are enlarged for display to fit a 256 x 256 quadrant. Each image pixel is enlarged to fill four display pixels in a 2 x 2 matrix. Consider that only 16 shades are viewable and an image pixel has a value of 28; the closest displayable grey levels are 16 and 32. By default all four display pixels would be mapped to 16. With the dithering scheme, one display pixel is assigned to 16 and the other three are set to 32; the average of the four is, therefore, 28. With this scheme in the order of 64 grey levels are viewable and most false contours are removed. A dithering technique is also used to write grey level images to the graphics printer. In this case each pixel is mapped to a 4 x 4 binary matrix and approximately 16 grey levels are viewable.

SPLIT LOOK-UP-TABLE

A useful capability is to be able to observe two images with different look-up-table maps at the same time. For example one image may consist of conventional intensity values which are best viewed with a grey scale while the second may be a vector field which is best viewed with a cyclic color sequence look-up-table. The split LUT facility is implemented by mapping one image to the pixel range 0 - 127 and the other to the range 128 - 277. In this way the two images, having seven active bits per pixel can be displayed with different halves of the LUT.

TWO AND THREE DIMENSIONAL GRAPHICS

It is useful to examine image data as a three dimensional graphics plot; this operation is supported by the software system and, currently, does not directly use the graphics features of the graphics display. The complete graph is generated on the host as a bit-plane image which is sent to the display in the usual way. A general two-dimensional graphics package is available for graphically displaying selected image information such as the profile of a specified image row (or column) of the image histogram. This package is also host based.

INTERACTIVE CURSOR AND GRAPHICS

The bandwidth between the graphics display and the host is too low for interactive pan and zoom of an image; however there is enough bandwidth to emulate an interactive cursor. To do this the LUT contents are modified so the least significant bit of all pixels is assigned for graphics use; ie., the display behaves as a 7-bit per pixel image display with a 1-bit graphics overlay. A cursor is displayed on the image, and is moved with single keystroke keyboard commands. Polygons can be drawn with the cursor in order to select regions of the image and text can be written on the overlay plane at the location of the cursor in order to annotate images. These two capabilities are implemented with the graphics commands of the professional graphics display.

INTERACTIVE LOOK-UP-TABLE MODIFICATION

Interactive programs have been developed which directly manipulate the contents of the LUT. A characteristic of many computer vision algorithms is that a threshold parameter must be set in order to distinguish between different regions of a feature image. An interactive program which involves the split LUT scheme permits a user to see both the feature image and the effect of changing a threshold parameter at the same time.

In addition to the above techniques a library of look-up-tables has been established. This library includes LUT contents for 8-bit grey scale, color spectrum, cyclic color, true color and 4-bit gray scale. Pairs of some of these combined for split LUT display are also available. The true–color LUT contents provided a limited capability for displaying color images. The 8–bit image pixels are divided into 3 bits for two of the primary colors and 2 bits for the remaining primary color.

THE SOFTWARE ENVIRONMENT

A software picture processing system (PPS) has been developed to aid research in image processing and computer vision. The system was designed to achieve the following goals: (a) provide users with an efficient set of programming tools for developing image processing algorithms, (b) provide easy access to special image peripherals and (c) make available a library of standard image processing functions. PPS programs are called Image Processing Modules (IPM's). They may all be used as filters and have a special PPS syntax for specifying parameters. Much of the flexibility of IPM's arise from the way file names may be specified.

PPS consists of a large set of primitive Image Processing Modules (IPM's) which are compiled programs that can operate on formatted image data. Formatted image data has a header which serves two functions: First it contains information for specifying the exact format of the image data and second, the header may optionally contain some descriptive information concerning the data. This feature can be very useful for maintaining some documentation of the data. The diverse range of applications indicated that a large set of data formats was desirable. However, designing an IPM to accept all possible data formats may be very difficult to achieve, will make all IPM programs larger and will made the addition of a new format to the system very difficult. This problem was solved by stating that IPM's need not be capable of accepting all data formats; however, they must check the format of all data and output an error message when a format problem is encountered.

A new format is simply added to the system by allocating a new format code number in the data header information. It is also useful to write a format–converting IPM which concerts from the new format to other formats and vice versa. Any formatting problems with existing IPM's may be solved by passing the data through the format–converting IPM.

IMAGE PROCESSING MODULES

Let the instance of a program which is running in a computer system be called a process. An IPM process is initiated by either an interactive user command or by another process. In either case, a set of parameters is passed to the IPM to specify where the data streams are to be located as well as other processing options. Data streams are conventionally directed to disk files; however, with UNIX, these streams may also be directed to devices or other processes, according to the parameters passed to the IPM.

There are several conventions associated with the parameters. In general, parameters are prefixed with a keyword, they are order independent, and defaults exist. An input data specifier is usually prefixed with if= and an output data specifier is prefixed with of=. When more than one data stream is used, the prefix includes the data stream number, e.g. a prefix if2= precedes the second input data stream specification.

The features of IPM input and output will be illustrated with two IPM's: hfft computes the FFT on all rows of an image and trans transposes an image. Hfft has an optional parameter –i which specifies that the inverse horizontal FFT of an image stored in fileA and store the result in a file called fileB. This operation is specified by the command:

```
hfft if=fileA of=fileB –i
```

In Unix, concurrent processes may communicate directly through interprocess data linkd=s called pipes rather than through files. Unix commands are executed by a command interpreter called the shell. The shell process can invoke other processes and can set up a pipeline of special processes called filters connected together by pipes. All IPM's are designed to behave as filters if an input or output file name specification is omitted.

For example, consider that a two–dimensional FFT of file A is required; this could be achieved with five commands as follows:

```
hfft if=fileA of=temp1
trans if=temp1 of=temp2
hfft if=temp2 of=temp3
trans if=temp3 of=fileB
rm temp1 temp2 temp3
```

480

The run command removes the temporary files temp1–temp3. This function may be restated to use pipes as follows:

```
hfft if=fileA/trans/hfft/trans of=fileB
```

where "/" specifies to shell that a pipe link is required. The use of interprocessor pipes has two advantages over the more conventional concept of running the four processes in sequence. First, the need for three explicitly stated intermediate disk files is avoided, and second, while the first process is waiting for disk I/O the other processes may be concurrently processing data; therefore, the throughput may be higher.

IMAGE DATA FORMAT

Image data in the PPS system may be formatted or unformatted. Unformatted data is simply a raw data file; the user must explicitly specify the format with the file name when processing this data with an IPM.

Formatted data has a header block of 512 bytes which precedes the raw data. The format of the data is specified by five integers in the header block:

x: x dimension (width)
y: y dimension (height)
z: number of bits per pixel
c: number of channels (for multispectral data)
f: format number (pixel format)

The header also contains two character strings of descriptive information called the title and description. The title may be changed by an IPM and many IPM's use the command character string for the new title. In this way, it is often possible to see what operations have been applied to a processed image by printing the title. The description may contain any information which the user would like to have maintained with the data.

Many data formats are available from single bitplane to multispectral floating point data, however IPM's usually only support one or two formats (8 bit unsigned integer is the most common). To alleviate this problem, some format converting IPM's are available. New formats may be easily added to the system by allocating a new format number (f).

DATA FORMAT NUMBERS

The data format in a PPS file is specified by the Format number (f parameter in the header); frequently the number of bits per pixel (the z parameter in the header) is also determined by the format number. The following data formats are the ones most commonly used:

Format number — Description

1. Byte data: this is usually unsigned. This is the format used for most images but it has many other applications also.

2. Integer data: this is signed 16-bit data. This is usually used when byte data does not have enough range.

3. Real data: this is in the 32-bit floating point format. Complex data, is usually represented by a format 3 image file with two channels; the first is real and the second is imaginary.

4. Double Precision data: this is usually 64-bit floating point data.

5. Signed byte: this is 8-bit byte data which has a sign bit; ie. numbers in the range −128 to +127 are represented.

6. Bit-plane data: this is a binary bitplane format; each byte contains a single bit from 8 consecutive pixels. When the z parameter is 1 then this is an efficient format for storing graphics information ie. images with 1-bit pixels. For this format the z parameter can in fact have any value; therefore pixels with an arbitrary number of bits can be specified.

In addition to the above image data formats a large number of feature formats are also supported for the computer vision applications.

PROCESSING UNFORMATTED DATA

Any IPM can accept unformatted data and also generate an unformatted result. For unformatted data input the size parameters of the image (x,y,z,c,f) must be specified after the filename; this is discussed in detail in the following section. The IPM may contain default values for unspecified parameters. For example, the following command will take the horizontal FFT of an unformatted 512 x 512 image prior and generate and unformatted result called outr:

```
hfft if=prior,f=3,x=512,y=512,z=32 of=outr,raw
```

The two-dimensional FFT for unformatted input and output could be specified by:

```
hfft if=prior,f=3,x=512,y=512,z=32/trans/hfft/trans of=outr,raw
```

The first hfft creates a header which is used by the following IPM's. The header is removed by the final transpose IPM.

PPS COMMANDS

The main groups of PPS commands are listed below. The number of commands in each group is given in parenthesis.

1. Display and Printer (10)
2. Data Format Conversion (21)
3. Image Pixel Operations (15)
4. Image Filters (22)
5. Image Transforms (15)
6. Feature Extraction
7. General Graphics (5)
8. Chain Code and 3-D Graphics (12)
9. Feature Vector Operations (28)
10. Image Generation (4)
11. PPS Header Operations (8)
12. Other (20+)

The UNIX (XENIX) capability of the invocation of processes by other processes is a very useful feature for this system. For example a composite IPM, called tdfft to do a two dimensional FFT operation by invoking a pipeline of four primitive IPM's as outlined previously can be implemented by a two line shell command file. The operation could then be specified by

```
tdfft if=fileA of=fileB
```

ie. with exactly the same syntax as if the IPM was a single process primitive IPM.

A second example is the display IPM which accepts almost any format image file and writes the image in a viewable form to fill the display screen (or a specified region or quadrant of the screen). To do this it first examines the format and size of the image; it then creates a pipeline of format converting IPM's with appropriate parameters to process the image file and write it to the display.

The PPS system is described in [2] and [3]. Extensions to enable shell programs to create multiple concurrent pipelines of IPM's are described, and distribution of IPM's over a computer network for faster processing is discussed.

CONCLUSION

An instructional environment for computer vision and image processing has been developed on an IBM PC/AT. This system is being successfully used in a number of courses. This has been achieved by porting a large software system from a research environment to the PC/AT. Almost all of this system has now been successfully ported. The PC/AT has proved to be a cost effective vehicle for developing image processing algorithms. In fact, when the research system is shared between several users, which is frequently the case, the PC/AT is a more convenient system to use. The professional graphics display is rather limited when used as an image display device; a number of techniques have been developed to circumvent some of these limitations.

REFERENCES

1. A. P. Reeves, "PPS–PDS Users Manual (version 2.5)", Cornell University, School of Electrical Engineering Technical Report, September, 1985.

2. A. P. Reeves, "A Structured Image Processing System" Proceedings of the 5th International Conference on Pattern Recognition, Miami Beach, Florida, December 1–4, 1980.

3. J. D. Bruner and A. P. Reeves, "An Image Processing System with Computer Network Distribution Capabilities," Proceedings of the IEEE Computer Society Conference on Pattern Recognition and Image Processing, June 1982, pp. 447–450.

AN INTERACTIVE HYBRID (SYMBOLIC–NUMERIC) SYSTEM APPROACH TO NEAR OPTIMAL DESIGN OF MECHANICAL COMPONENTS

Presenter: **David A. Hoeltzel** *Columbia University*

Coauthors: **David A. Hoeltzel** *Columbia University*
Wei–Hua Chieng *Columbia University*

ABSTRACT

A hybrid (symbolic–numeric) system, referred to as OPTDEX, (OPTIMAL DESIGN EXPERT) for the optimal design of mechanical components and systems has been developed. The system is written in Golden Common LISP and IBM Professional (Ryan–McFarland) FORTRAN for execution on the IBM PC/AT microcomputer. Graphical output has been implemented using the Graphical Kernal System (GKS) standard. This microcomputer–based implementation makes the system particularly attractive as an easily accessible, low cost engineering analysis and design tool.

Experience with the system indicates that the time required to achieve, at least partially optimized engineering design solutions, is similar to that which may be expected with standard, nonoptimization–based, microcomputer computation. Any added computational time may be justified and subsequently offset by increased long term design efficacy.

The OPTDEX protocol (figure 1) assumes a modular form whereby each level can be modified, updated and enhanced independently of the others to accommodate various design philosophies and the subdivision of large scale design problems. A "design cell" approach has been adopted which has the capability of addressing the design of various mechanical components and systems. The current version of the OPTDEX design cell library, which is undergoing revision and expansion, includes speed reducer, bearing, coupling and shaft design capabilities.

This modular structure and generalized design cell approach, which underlies the OPTDEX system, establishes the basis of a formalized methodology for mechanical engineering design which may be extended to include other design related disciplines as well. For example, with the addition of appropriate

485

design cells, the system can be configured for VLSI circuit design in electrical engineering, scheduling and job routing in industrial engineering and structural design in civil engineering.

INTRODUCTION

Presently, optimization in engineering design is an all too underutilized means of achieving significant engineering design refinements. This is primarily due to the high level of sophistication required to apply optimization techniques successfully as well as the general lack of availability of easily implementable design optimization codes. In short, an expert is typically required to formulate and guide a design optimization problem through to successful fruition.

In an effort to remedy these shortcomings attempts are being made to simplify the application of nonlinear optimization algorithms in engineering design practice. Recently, a number of investigators from fields such as mechanical engineering, Jha [1], systems engineering, Betts [2], and applied mathematics have begun to consider the problem:

Artificial Intelligence

+

Mechanical Design

+

Optimization Method

=

Expert System For Optimized Mechanical Design

While progress in expert systems research has been made in several specialized mechanical design areas such as linkage synthesis [3] and air cylinder analysis [4], a general underlying theory of what an expert system should be within the context of a mechanical design problem remains unresolved by the artificial intelligence in design research community. Rasdorf [5] and Brown et al. [6] offer some important insight into this problem. Betts [2], emphasizes the need for a careful interfacing of needs and requirements between optimization algorithms and engineering systems to which these algorithms are applied. It is here, where mismatch between theory and practice occurs, that advances must be made if optimization is to be readily understood, implemented and applied to "real world" design problems.

From a systems engineering standpoint the problem appears to be one of determining an appropriate hierarchy of design, optimization and artificial intelligence factors for problem formulation as well as the effective generation (implementation) of numerical procedures (algorithms) for problem solution.

The OPTDEX system affords a structured approach to engineering design optimization which can span numerous engineering specialties. The system offers a general methodology for design optimization problem formulation and solution.

DESIGN OPTIMIZATION

The application of optimization methodology to system design can result in remarkable results both in terms of system efficacy and design flexibility. Various computer programs have been specifically developed for the application of nonlinear programming algorithms to engineering design optimization problems. Several of the more widely known and distributed design optimization codes include ADS by Vanderplaats [7], OPT and BIAS by Gabriele, Root and Ragsdell [8,9] and OPTIVAR by Siddall [10]. These programs can be effective in solving design optimization problems provided the design problem has been property posed.

However, the basic underlying philosophy of nonlinear programming algorithms is that no one strategy can be relied on to be best for any given problem. As a result, there is always some complicated best match between problem and optimization strategy [11]. Expert optimizers have the capability to switch from one method to another in an attempt to speed the convergence process. The expertise required to perform this switching can be such that most novice optimizers will find it quite difficult, if not impossible, to perform on a timely basis. The EXADS program [12] represents a beginning in the application of an expert system front end to a general purpose design optimization code.

It is in the formulation of problems and selection of methods and strategies, that knowledge-based expert systems can speed the design optimization process.

EXPERT SYSTEMS

An engineering design expert system is one that handles real-world, complex design problems requiring an expert's interpretation and solves these problems using a computer model of expert human reasoning, reaching the same conclusions that t he human expert would reach if faced with a comparable problem [13]. If a computer program is to perform a design task in a manner similar to that of an expert it requires a large amount of knowledge. As a result, such systems must be organized in a structured fashion if confusion is to be avoided and efficiency is to be achieved. Production systems [14] represent a method of organizing expert system programs into three basic groups [15] which include:

1. Factual knowledge – This knowledge represents a specific case which is acquired through dialogue with the system user to establish facts about thecurrent problem.
2. Procedural knowledge – This knowledge is acquired in advance from a domain (engineering design) specialist and forms the foundation of a knowledge base.
3. Control knowledge – This knowledge contains control strategies which enable the system to try alternatives and to deal with failed attempts.

The production system approach adopted in the current design optimization system contains a (1) design data base, (2) list of design oriented production rules and (3) control schemes for choosing the appropriate design production rules to apply to the data base.

HYBRID SYSTEMS

When numerical (analysis system) and symbolic (expert system) computations are in some way coupled, the resulting program may be classified as a hybrid system or knowledge-based system [16]. Hybrid system approaches are typically found in engineering analysis applications where an expert system is required to set up the data which defines the problem, execute the analysis programs and retrieve the output for subsequent evaluation.

Design optimization problems are ideally suited for formulation as hybrid problems with a symbolic processor selecting an appropriate optimization method and a numeric processor performing the actual optimization analysis.

The OPTDEX system operates in this fashion and may be classified as a hybris system.

THE OPTDEX SYSTEM

OPTDEX is currently configured as a hybrid power–transmission–assembly design framework which emulates the intelligence of a human designer, manages the domain design knowledge and controls the numerical processing.

Based on observations of the problem solving strategies of the human designer and modeling of the taxonomy and topology of the design process it appears that design processing is structured, gradually from the symbolic to the numerical, from general knowledge representation to a more specific analytical representation. The logical flow of design information and processing within the OPTDEX system contains five major subdivisions (Figure 1):

1. Top Level Design Manager (TLDM) – The key task of this level is the representation of design methodology, taking advantage of a knowledge–based expert system (KBES) approach and formulating design problems heuristically.

The TLDM is designed to perform as a planning program based on the knowledge contained within the design rules. Figure 2 depicts a basic planning example which might occur in the TLDM. The knowledge is encoded in the form of antecedent–consequence pairs of IF–THEN rules:

```
RULE_TLDM_01
if      ?x is_an out_port and has_effect intermittent motion
        and ?p is_a point_in_design_space directly_connected
        with ?x and ?p can_not has_effect intermittent_motion and
        function: enough_space_for a_cell clutch in_between
                      [(position_of ?x) (position_of ?p)]
then    create_point named +ip
        add cell clutch to production_queue position
        function: insert_in_region
                  [(position_of ?x) (position_of ?p)]
                  local_boundary_conditions
                  input is ?p and output is ?x
                  <This function drives the inference engine
                  to search the boundary conditions of ?x and
                  ?p, then binding to the local boundary
```

```
                list of clutch +ip>
delete          the influence intermittent_motion form
                power_line (input_port ?intermittent_points
                                          -list ?x) and
store_protect (need clutch named ?ip) end_of_now to
                begin_of cell_design
```

Aside from the planning rules, protection and replanning rules are also included. A noteworthy point concerning the data base is data dependency.

After the replanning project is complete, the program must delete all relatives of the failed design plan. Data dependency has been implemented using a sequential numbering of plans. Once a deletion occurs, the inference engine can easily search for redundant data and delete it.

When the design process passes through the TLDM is breaks down the global boundary conditions into local cell boundary conditions which become queries in a production queue for the CLDM. As an example:

```
(cell      'coupling
           '(position 1.2 3.2 4.1)      * specified according
                              to user defined origin and
                              units *
           '(input_torque 499 lb-in)

           '(speed 250 rpm)

           .. .. ..
```

2. Cell Level Design Manager (CLDM) – Representing real-world design problems is the purpose of this level. This level contains the symbolic optimization capabilities, which are based on domain knowledge, for the formation of cells to be designed in an optimal sense. The CLDM is a heuristic search program (figure 3).

3. Optimization Level Design Manager (OLDM) – Contains existing nonlinear programming algorithms (mathematical optimization methods) which perform the actual numerical optimization for a discrete design. For a specific design problem the program first goes into the design equation data base (figure 4A) and searches for all applicable equations. The selection of design equations is based on a hierarchical data structure. The optimization application programs

(FORTRAN) require the design objective functions and the design constraints. The design objective is assigned as:

OBJ = weighting_factor_1 * obj_1 + weighting_factor_2 * obj_2 +...

When different objective functions are assigned by the user it is possible to change only the weighting factor of the corresponsing design objective.

For some available nonlinear programming codes, such as ADS, the programming methodology is quite regular. It is therefore a straightforward process to translate the design equations into FORTRAN code. A program (ADSC) has been developed for this purpose using Golden Common LISP (figure 4B).

After implementing the optimization programs, the next step is to establish the best optimization method for the problem. In order to avoid exhaustive testing of all the possible algorithms (although some optimizers believe that this is necessary) an expert system (figure 4C) has been implemented which judges the convergence critieria and controls the optimization switching process.

As an example:

```
if      last_strategy is ?x and last_running exists error
        or end_of optimizer_testing and end_of_1-D_search
        -testing or not
        function: improvement_in_the_last_three_test
then    new_strategy is
        function: get_the_highest_priority_so_far
```

Additional information includes rules to judge step length which determin convergence and control the optimization process.

4. Graphics Level Manager (GLM) – Displays the final result of a design in a standard format. This graphics process differs from general graphics tools in that its totally controlled by the OPTDEX system (not the user).

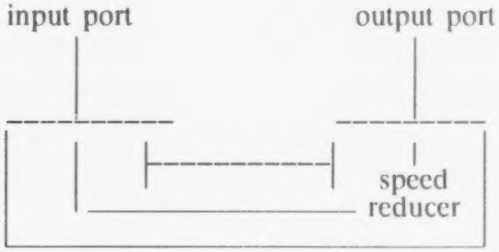
The GLM contains a data structure typical of that found in finite element programs, that is, a sequential list of X,Y and Z coordinates which define the boundaries of the mechanical component or system. The GLM acquires parameterized design data, for the specific component, from the design data base enabling it to plot the component. The GLM then displays both the design space and the component according to three views (top, front, side) and a hidden line plot. Dimensional scale factors and component dimensions are also provided. Figure 5 depicts a representative display produced by the GLM.

5. Learning Manager (LM) – The capability to understand the designer's suggestions are represented here, since the top level designer only includes rules of thumb implemented during the development stage of OPTDEX. For special applications there may be rules requested by the user which do not currently exist in the data base. The system needs to understand how to perform a new design process in accordance with the user's definition of this new process. In the absence of learning ability the system can only think as an expert's system and not as an expert system.

Design learning, as it appears in the OPTDEX system, differs from the process of pure learning in that the primary user of the system will be a noveice (nonexpert). The ability of the LM to acquire knowledge about the design process will therefore be limited by the inexperience of the user.

Learning can be divided into two parts, the first is a failure driven, the second is new concept driven. Failure driven learning is triggered by the occurence of a failure. As a result of this failure, the LM will automatically adjust the rules of the TLDM and retrieve a new path for redesign processing.

An example of the new concept learning process might be the successful design of a power transmission assembly, that is a design in which no constraints have been violated. The result might appear as:



However, the user has found that this design is not properly balanced (left to right) and requests a balanced design. If this is a failure driven learning mechanism then, based on the fact that the LM holds no failure rules for an unbalanced design, appropriate action cannot be taken.

An alternative way of applying concept learning is through "specialization" or "generalization" [17]. When a situation arises involving a novice user, who introduces a new concept, generalization rules will not be produced since the new concept represents new data which is unable to match any data already in the data base. If all negative instances are assigned then many over-specified rules will be created permanently affecting some important design rules in a detrimental way. In order to alleviate this problem, the idea of learning authority is suggested, whereby the origin of any new learning concept must be determined. For novice users, the revised rules will only remain in effect temporarily, being automatically erased at the completion of a design session.

DESIGN EXAMPLE USING OPTDEX

A design production example is presented which demonstrates processing in the OPTDEX design optimization system (Pages V−53 to V−56). A design space with its input and output ports is depicted. The TLDM first determines if the user's input is correct, then according to the input, constructs the design plan and produces queries to drive the CLDM. The productions are in the form:

```
(cell <cell_instance>
          <cell_boundary_condition_1>
          <cell_boundary_condition_2>
          ........                    )
```

When the process reaches the CLDM it searches the inheritance net to find the best choice for each design cell. If the cell coundary conditions are too general then it will not be possible for the CLDM to make a clear choice and the system will proceed to failure mode and wait for the user to reassign the specifications. If no errors occur in this level then the process woll proceed to the OLDM and search for the design equations and best nonlinear programming method for numerical optimization.

From this example it can be seen that the hybrid system passes control from symbolic processing to numerical processing in such a way that the numerical processing assumes a dominant role.

CONCLUSION

Optimization theory has and will continue to represent an important aspect of engineering design. However its usefulness depends, to a great extent, on its successful implementation for domain specific problems. The prototype of a mechanical problem solver (OPTDEX) represents an approach to the methodological representation and optimum solution of mechanical design problems.

Experience with the OPTDEX system has demonstrated that significant enhancements to the outcome of the design process can be realized with relatively little additional computational effort over that with nonoptimization–based computational design time.

Planned refinements/additions to the OPTDEX system include:

1. Advanced learning manager for advanced user instruction.
2. Direct three–dimensional geometric data input for reverse modeling.
3. Addition of other components and systems.
4. More sophisticated Optimization Level Design Manager to reduce time search.
5. More sophisticated Graphics Design manager for improved visual function evaluation and mechanical system description.

ACKNOWLEDGEMENTS

This research was supported by grants from the IBM and NCR Corporations.

REFERENCES

1. Jha, N. K., Engineering Optimization and Expert Systems, Proc. 1985 ASME Int. Comp. In Eng. Conf., pp 97–101.

2. Betts, J. T., Fronteirs in Engineering Optimization, ASME Journal of Mech., Trans., and Auto. In Design, June 1983, Vol. 105, pp 151–154

3. Thompson, T. R., Riley, D. R., Erdman, A. G., An Expert Systems Approach To Type Synthesis of Mechanisms, Proc. 1985 ASME Int. Comp. In Eng. Conf., pp71–75.

4. Kumar, A., Kinzel, G. L., Singh, R., A Preliminary Expert System For Mechanical Design, Proc. 1985 ASME Int. Comp. In Eng. Conf., pp 29–35.

5. Rasdorf, W. J., Perspectives On Knowledge In Engineering Design, Proc. 1985 ASME Int. Comp. In Eng. Conf., pp 249–253.

6. Brown, D. C., Chandrasekaran, B., An Expert System For Mechanical Design: A Progress Report, Proc. 1984 ASME Int. Comp. In Eng. Conf., pp 343–344.

7. Vanderplaats, G. N., ADS – A FORTRAN Program for Automated Design Synthesis, NASA Contract Report 172460, 1984.

8. Root, R. R., Ragsdell, K. M., BIAS: A Nonlinear Programming Code in FORTRAN IV – Users Manual, Purdue Research Foundation, West Lafayette, Indiana, 1978.

9. Gabriele, G. A., Ragsdell, K. M., OPT: A Nonlinear Programming Code in FORTRAN 77, 1984, Design Optimization Laboratory, University of Missouri, 1982.

10. Siddall, J. N., OPTIVAR: Designers Optimization Subroutines, McMaster University, 1982.

11. Siddall, J. N., Optimal Engineering Design: Principles and Applications, Marcel Decker, Inc., 1982.

12. Rogers, J. L., Barthelemy, J. F., An Expert System for Choosing the Best Combination of Options in a General–Purpose Program For Automated Design, Proc. 1985 ASME Int.Comp In Eng. Conf., pp 255–260.

13. Weiss, S. M., Kulikowski, A. A., A Practical Guide to Designing Expert Systems, Rowman & Allanheld Publishers, 1984

14. Nilsson, N. J., Principles of Artificial Intelligence, Tioga Publishing Co., 1980.

15. Forsyth, R., Expert Systems Principles and Case Studies, Chapman and Hall, Ltd. 1984.

16. Zumsteg, J. R., Flaggs, D. L., Knowledge-Based Analysis and Design for Aerospace Structures, Applications of Knowledge-Based Systems to Engineering Analysis and Design, ASME pub. AD-10, pp 67-80, 1985.

17. Charniak, E., McDermott, D., Introduction to Artificial Intelligence, Addison Wesley, 1985.

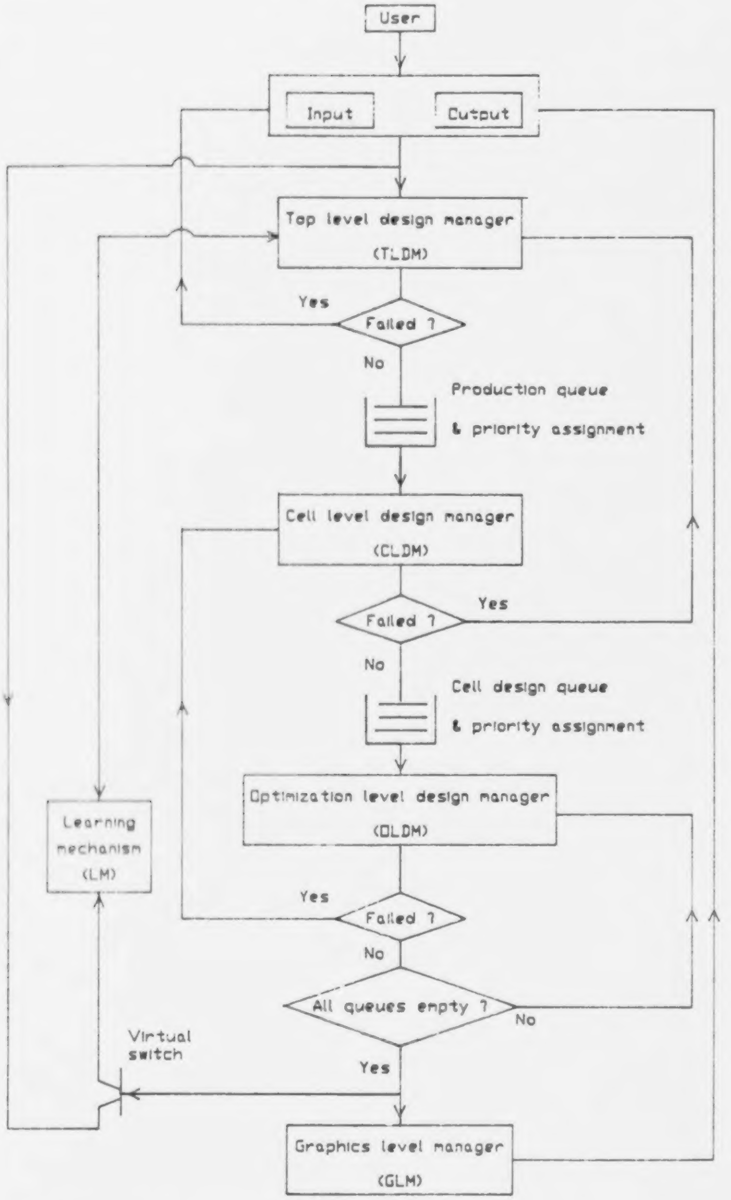
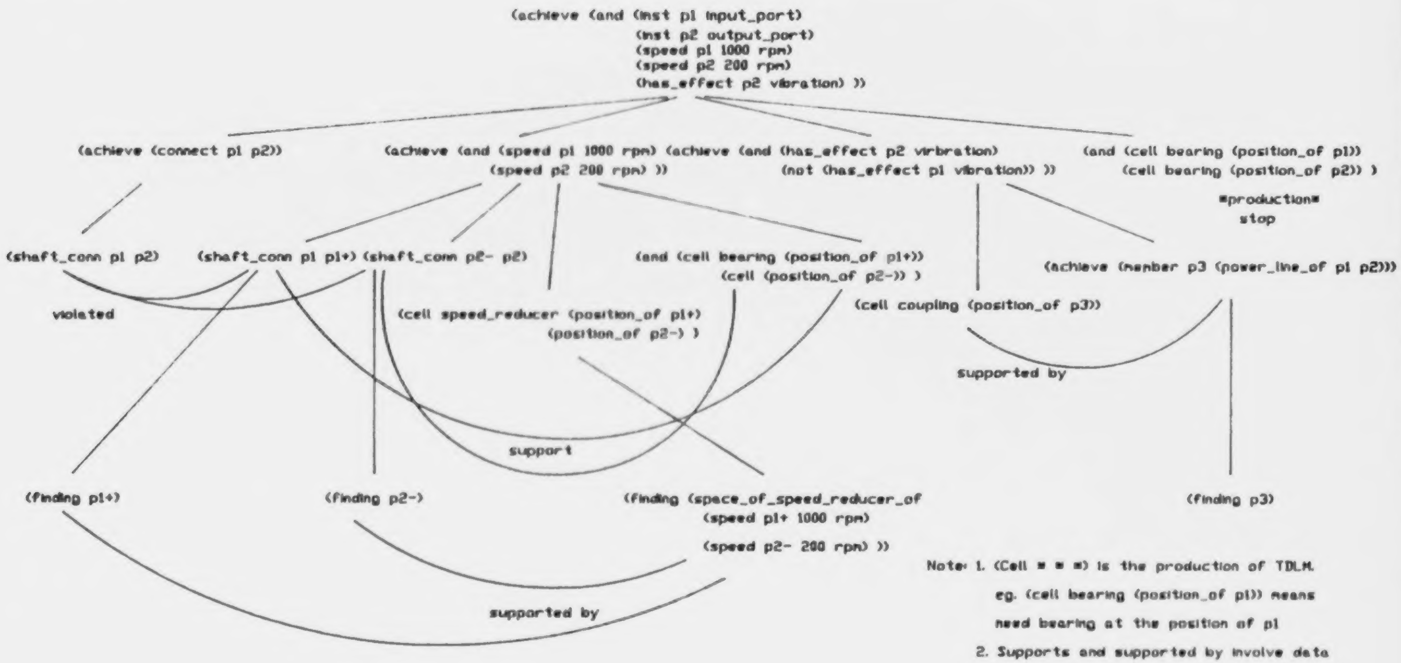
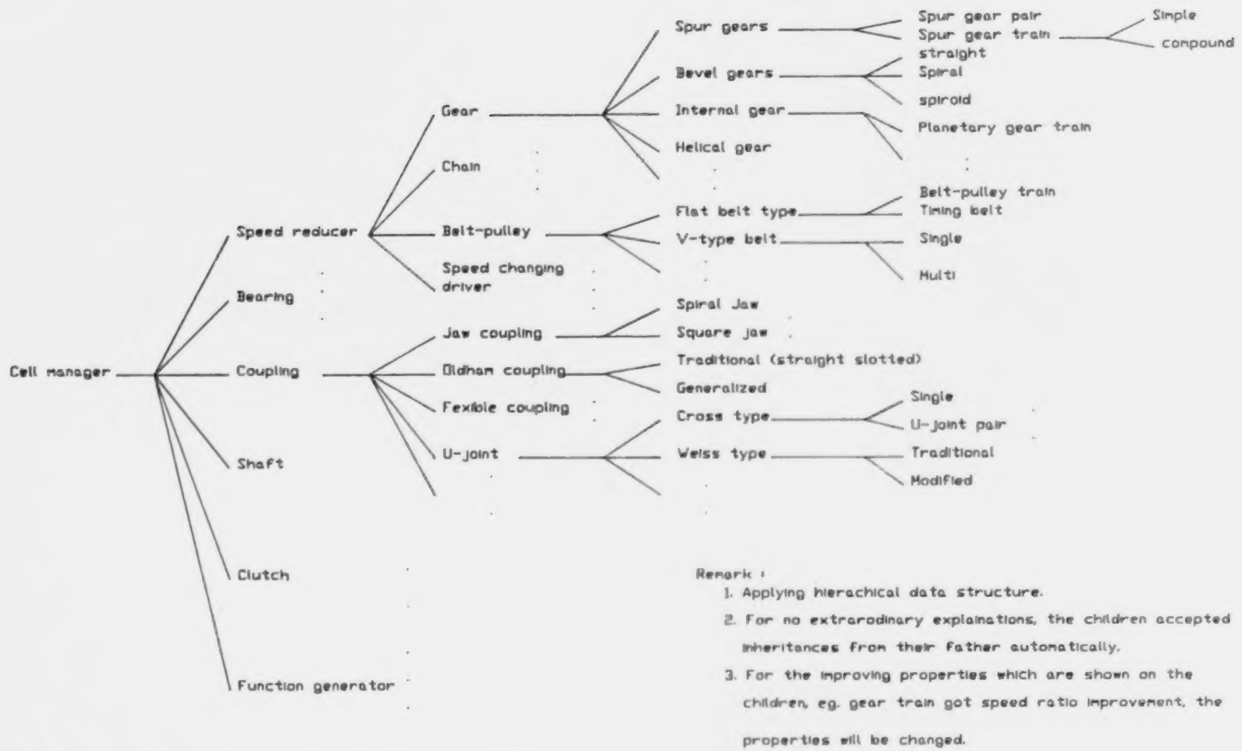


Figure 1. Flow control in the OPTDEX system.


(achieve (and (inst p1 input_port)
(inst p2 output_port)
(speed p1 1000 rpm)
(speed p2 200 rpm)
(has_effect p2 vibration)))
(achieve (connect p1 p2))
(achieve (and (speed p1 1000 rpm)
(speed p2 200 rpm)))
(achieve (and (has_effect p2 virbration)
(not (has_effect p1 vibration))))
(and (cell bearing (position_of p1))
(cell bearing (position_of p2)))
production
stop
(shaft_conn p1 p2)
violated
(shaft_conn p1 p1+)
(shaft_conn p2- p2)
(and (cell bearing (position_of p1+))
(cell (position_of p2-)))
(cell speed_reducer (position_of p1+)
(position_of p2-))
(cell coupling (position_of p3))
(achieve (member p3 (power_line_of p1 p2)))
supported by
support
(finding p1+)
(finding p2-)
(finding (space_of_speed_reducer_of
(speed p1+ 1000 rpm)
(speed p2- 200 rpm)))
(finding p3)
supported by

Note: 1. (Cell * * *) is the production of TDLM.
eg. (cell bearing (position_of p1)) means need bearing at the position of p1
2. Supports and supported by involve data dependency
3. When simply finding is failed ,the leave function finding also involve planning .

Figure 2. A basic planning example in the TLDM using a production system.



Remark :

1. Applying hierachical data structure.
2. For no extrarodinary explainations, the children accepted inheritances from their father automatically.
3. For the improving properties which are shown on the children, eg. gear train got speed ratio improvement, the properties will be changed.

Figure 3. Inheritance net for database of CLDM.

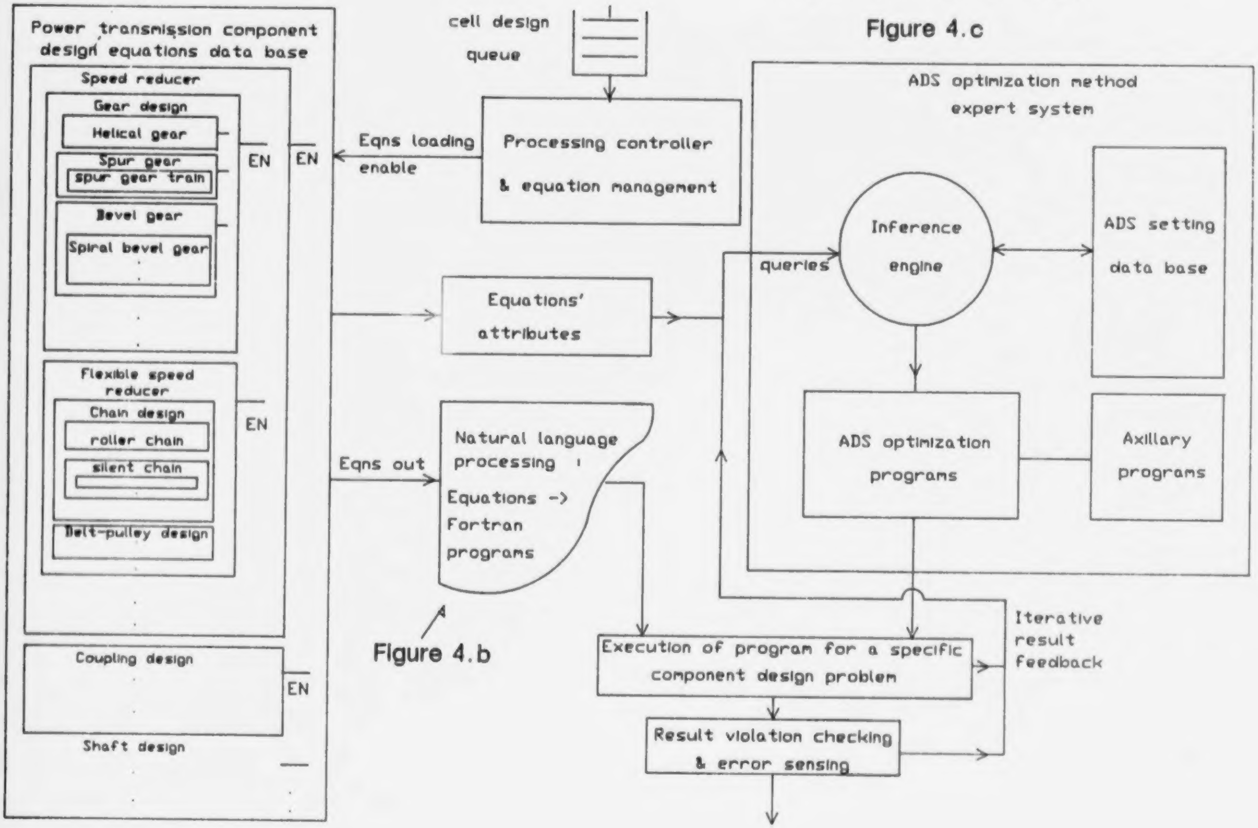
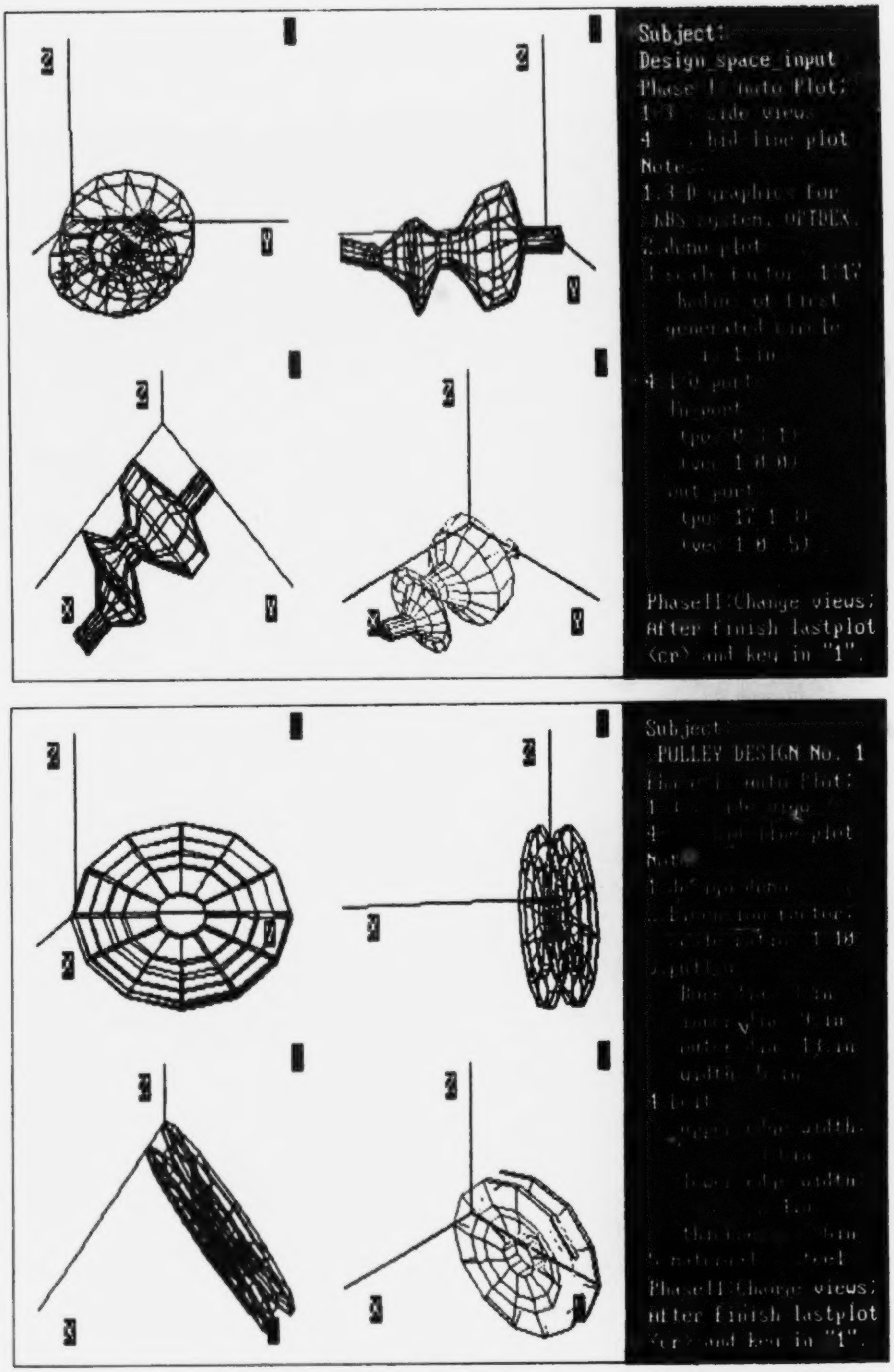

Power transmission component design equations data base
Speed reducer
Gear design
Helical gear
Spur gear
spur gear train
Bevel gear
Spiral bevel gear
Flexible speed reducer
Chain design
roller chain
silent chain
Belt-pulley design
Coupling design
Shaft design
EN
EN
EN
EN
cell design queue
Eqns loading enable
Processing controller & equation management
Equations' attributes
Eqns out
Natural language processing :
Equations -> Fortran programs
ADS optimization method expert system
Inference engine
queries
ADS setting data base
ADS optimization programs
Axillary programs
Execution of program for a specific component design problem
Result violation checking & error sensing
Iterative result feedback

Figure 4.a

Flow control of OLDM

Figure 4.c

Figure 4.b

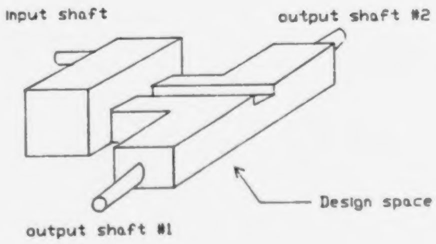


```
Subject:
Design_space_input
Phase I:Auto Plot;
1-3 : side views
4   : hid line plot
Notes:
1.3-D graphics for
 KBS system, OPTDEX.
2.demo plot
3.scale factor: 1:17
   Radius of first
  generated circle
     is 1 in
4.I/O port:
  In_port
   (pos: 0 1 1)
   (vec: 1 0 0)
  out_port
   (pos: 17 1 1)
   (vec: 1 0 .5)

PhaseII:Change views;
After finish lastplot
<cr> and key in "1".
```

```
Subject:
 PULLEY DESIGN No. 1
Phase I:Auto Plot;
1-3 : side views
4   : hid line plot
Notes:
1.demo [illegible]
2.Reduction factor:
  scale factor: 1:10
3.pulley:
   Bore dia:   1 in
   inner dia:  3 in
   outer dia: 13 in
   width:      5 in
4.belt:
  upper edge width:
           [illegible] in
   lower edge width:
           [illegible] in
   thickness: [illegible] in
5.material:   steel
PhaseII:Change views;
After finish lastplot
<cr> and key in "1".
```

Figure 5. Graphical output from GLM (using IBM color Jet printer).

```
TLDM > user_I/O_ports_and_design_space_specification
```

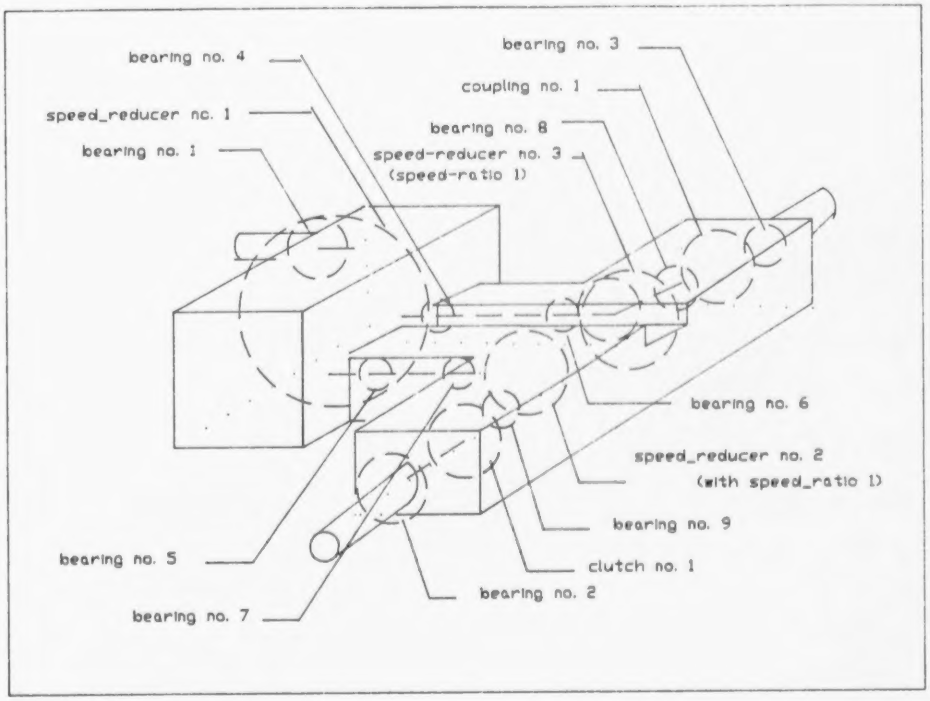


```
I/O port specification :

Input port :                  Ouput port #1 :                Output port #2 :

1. motion type : angular      1. motion type : angular       1. motion type : angular
2. Speed : 2400 rpm ccw       2. Speed : 80 rpm ccw          2. Speed : 400 rpm cw
3. Torque : 100 lb-in         3. Torque : 2000 lb-in         3. Torque : GE. 220 lb-in
4. Position : (0.5 ,0.5, 1.5) 4. Position : (4, 5, 0.5)      4. Position (0., 3.5, 5.)
  vector : (0, 1, 0)             vector : (1, 0, 0)             vector :(-1 , 0, 0)
  Unit : 3 in                 5. Effect : intermittent       5. Effect : vibration
                                          motion
Note: the vector_of in_port points into_design_space but

     the_vector_of out_port points out_of_space
```

```
TLDM > ** specification error **

       1. The summation of output power is greater than of input

       NIL

User response > (spec out_port_1 speed 1500 rpm)

TLDM > pass specification_checking

TDLM > suggestion_request:

       1. The speed_reducer near in_port suggest input position_of_it ?

       2. The speed_reducer is_not near out_port_1 and impossible_connect_them_directly

         suggest (p 1) a_intermediate_point ?

       3. The speed_reducer is_not near out_port_2 and impossible_connect_them_directly

         suggest (p 2) a_intermidiate_point ?

       NIL
```

-- after user's appropriate input --

```
TLDM >  *production-list*
         ( (cell bearing no. 1 (pos .5 .5 1.5 unit: 3 in) (speed 2400 unit: rpm)
            (load radial: 100 thrust: (defend_on speed_reducer no.1) unit: lb-in) )
          (cell bearing no. 2 (pos 4. 5. .5 unit: 3 in) (speed 80 unit: rpm)
           (load radial: 1500 thrust: 750 unit: lb-in) (intermittent_motion yes) )
            .    .    .
          (cell speed_reducer no. 1 (pos center: 2. 1. 1. unit: 3 in) (no._of_output 2)
           (speed_ratio 30 6) (max_speed 2400 unit: rpm) (max_torque 1000 unit: lb-in)
           (power 4 unit: hp) (efficiency .8) (cost_limit no) (noise_limit no)
           (shaft_offset 2.1 0.9 unit: 3 in) (shaft_misalignment 0 unit: deg)
           (power_density 0.333 unit: hp/in**3) (axillary_condition no) )
          (cell speed-reducer no. 2 (pos center: 3. 5. 0.5 unit: 3 in) .....
          (speed_ratio 1) ... (shaft_misalignment 90 unit: deg) .....)
           .   .     .        .
         (cell coupling no. 1 (pos 0.5 5. 0.5 unit: 3 in) (speed 400 unit: rpm)
          (torque (> 220) unit: lb-in) (speed_variaition default) (cost_limit no)
          (shaft-misalignment (> 10) unit: deg) (vibration yes) (shaft_offset no) )
         (cell coupling no. 2 . . .   . . )  .  .  .
         (cell clutch no. 1 (pos center 3.5 5. .5 unit: 3 in) (torque 1500 unit: lb-in)
          (speed 80 unit: rpm) (power 2 unit: hp) (noise_limit no) (cost_limit no)
          (frequency default) (effective_damping_ratio default) )
           .      .     .           )
TLDM >  pass top_level_design_manager
TLDM >  go_to cell_level_design_manager
```



A scheme iterpretation of productions which is not shown on OPTDEX

```
                    CELL LEVEL DESIGN MANAGER

CLDM > speed_reducer no. 1 : the_bset_choice is

       spur_gear train with euclidean_distance -30

       produce : (thrust_load_of_neighbour_bearing 0)

CLDM > bearing no. 1 : the_best_choice is

       radial_contact roller_bearing with euclidean_distance -35

       straight roller_bearing with euclidean distance -35

       produce : NIL

CLDM > bearing no. 2 : the_best_choice is

       Double_row spherical_roller_bearing with euclidean_distance -10

       produce : NIL

       adding_boundary_conditions : (add bearing no. 2 (material composite_material))
```

```
CLDM > coupling no. 1 : the_best_choice is

      universal_joint with euclidean distance -20

      produce : (and (not (spec ?x vibration)) (shaft_connect ?x coupling no. 1)

                    (not (inst ?x out_port_1)) )

CLDM > clutch no. 1 :  *** failed ***

      no enough_information_to_reach_a_best_choice
```

Note: At this point the CLDM has failed and no further processing will proceed until new boundary conditions come in, the reason for 'no futher prceeding is after user changed (or simply adding) boundary conditions the whole productions of TLDM will also be changed, so the old productions can't help the final result ,but this is still a tricky problem for paralle processing architecture, since there must be some old productions which independent of the failure. To this point we may discuss on the later paper.

-- after user's appropriate response, design process goes to --

```
OLDM >spur_gear_train design_equations :

      2 static_eqn  4 dynamic_eqn  2 kinematic_eqn are available

      Obj = 0.95 * obj_on_space_occupation + 0.05 obj_on_costing + 0

      equation_attributes: (function_evaluating_costing much)

                           (diferentiable_to_order 2)  (function_highly_nonliear yes)

      OLDM > the_first_method (ADS (istrat 0) (iopt 4) (ioned 7) )

      OLDM > beginning spur_gear_train design
```

CLOSED–LOOP DYNAMIC TESTING CONTROL USING AN IBM PC/AT

Presenter: **Jorge B. Sousa** *University of California, Berkeley*

Coauthors: **Jorge B. Sousa** *University of California, Berkeley*

Carl L. Monismith *University of California, Berkeley*

ABSTRACT

A research project to determine properties of pavement materials required the excitation of hollow cylindrical specimens using two independent sinusoidal loads with frequencies of up to 30 Hz.

An IBM PC/AT equipped with a Metrabyte DASH16 Data Acquisition Board was used to directly control two MTS Hydraulic Servorams.

Menu–driven software was developed, taking advantage of DMA channels, so that data acquisition, wave form generation, and closed loop control could simultaneously take place at rates above 5000 samples/sec/channel.

A post–processing menu–driven program converts raw data into graphs and parameters representative of dynamic material properties.

Based on the same approach, existing laboratory tests, routinely executed as part of class work, were redesigned, thus transforming data acquisition, equipment control, and data processing from tedious, time–consuming tasks into stimulating experiments.

INTRODUCTION

Recent developments in microcomputers, such as increased speed and memory capabilities, have opened new avenues in data acquisition, process and equipment control, automation, and data analysis. These innovations have proven a valuable tool in both research and education. A current research project made use of microcomputer capabilities in order to develop a new testing system that determines dynamic properties of pavement materials. In an educational environment, a routinely assigned laboratory experiment was restructured with the aid of a microcomputer, facilitating students' comprehension and visualization of the concepts being taught.

A RESEARCH PROJECT - DYNAMIC LOADING SYSTEM

The main objectives of this research are to determine the dynamic properties of some representative pavement materials and to incorporate them in an analytically-based method which will evaluate the effects of wheel loads. To make the analysis with appropriate material parameters, a laboratory test program determining material properties under realistic stress paths and with simulated in situ stress conditions was implemented. Using this program elastic, non-elastic, and damping parameters will be defined for fine-grained soils representative of sub-grade materials, granular materials representative of aggregate base courses, and asphalt-concrete mixtures.

For this purpose, a testing device was fabricated to test both hollow cylindrical and solid cylindrical specimens. The apparatus, henceforth referred to as the Dynamic Loading System, is capable of applying torsional and axial loads with different frequencies (up to 30 Hz) either independently or simultaneously, thus enabling the rotation of the principle axes of stress to follow predefined stress paths.

Current advances in microcomputer technology provide a relatively inexpensive way to develop apparatus where the accuracy and power of feedback closed-loop control hydraulic systems is combined with the flexibility, automation, and speed capabilities of digital equipment. However, for the purposes of this project, specially-developed software was necessary since all commercially-available programs were found to be inadequate.

APPARATUS DESCRIPTION

SPECIMEN DIMENSIONS AND PRESSURE CHAMBER

An axonometric view of the hollow cylindrical specimen and the adjacent apparatus is shown in Fig. 1. The hollow cylindrical specimen is 18" (45.7 cm) high, with a 9" (22.8 cm) external diameter and a wall 1" (2.54 cm) thick. The specimen is contained between a membrane and cap and base rings made of hard anodized aluminum. Between the load cell and the cap ring a top plate can be placed when different interior and exterior confining pressures are required. The apparatus can also be used with solid cylindrical specimens 9" in diameter and 18" high. Solid specimens are necessary when granular materials, with large particle aggregates, are tested.

TEMPERATURE CONTROL

Temperature control is achieved by circulating the pressurized air contained within the cell through a copper serpentine located in the interior of an insulated box. Heat is transferred to and from the serpentine by heating and cooling systems controlled and monitored by a temperature-sensitive sensor. The feedback censor is placed inside the return pressure line near the top chamber plate.

SLIPPAGE FREE SURFACES

In order to insure adequate friction between the specimen and the cap and base rings, eight 0.5" (1.27 cm) lugs are mounted on the rings. Another advantage in using lugs is that, under certain loading conditions, they do not require any additional means of load transfer to the specimen. However, if complete reversal of torsional stress and/or axial traction are required, then epoxy glue or hydrostone must be used.

DEFORMATION MEASUREMENTS

Measurements of specimen deformation are made using LVDTs. The LVDTs used are in constant contact, due to spring-loaded gauging heads, with a target mounted on the specimen. The target can be positioned anywhere on the specimen by using either fastening screws, when asphalt mixtures are being tested, or a clamping ring surrounding the specimen, when testing fine-grained soils.

However, LVDT's are not recommended for frequencies above 20 Hz. For such high frequencies, accelerometers should be used since they can be mounted directly on the specimen, thus eliminating possible errors due to the vibration of the mounting device.

LOADINGS SYSTEM

The Dynamic Loading System is designed in such a way that the loading mechanisms are mounted on a heavily reinforced frame beneath which the specimen is placed. Both torsional and vertical loads are transferred through the same shaft to the wheel-shaped load cell.

Vertical loads are applied by a vertically–mounted 10 Kip MTS servoram with a 6" stroke. Torsional loads are applied by a horizontally–mounted, 6" stroke, 2 Kip MTS servoram. The latter, when connected to a torque arm 7" (17.78 cm) in length, provides up to a 14 Kip in torsional moment to the center shaft (see Fig. 2).

A Rotolin linear and rotary ball bearing, with a basic capacity of 2.4 Kip and a total linear travel of 1", allows the coupling of both loads through the same shaft. Decoupling of the vertical movement of the torque arm from the horizontal movement of the horizontal servoram is made possible by two spherical bearings. The Rotolin bearing limits the vertical travel of the shaft to 1", and the spherical bearings limit the angle of the torque to +/– 15x.

The wheel–shaped load cell was constructed with eight spokes, each in the form of a rectangular prism. As load transfer mechanisms, each of the spokes are assumed to function as beams connecting the inner hub, where the center shaft is fastened. Vertical and torsional loads can then be measured independently since strain gauges are mounted on the center lines of alternate faces of the beams.

DATA ACQUISITION AND EQUIPMENT CONTROL

The development of microcomputers, together with the development of interfaces between digital and analog equipment, now provide a relatively inexpensive way to build sophisticated apparatus with which tasks like wave form generation, data acquisition, equipment control, and data processing can be accomplished almost simultaneously.

To successfully perform all the above tasks, a microcomputer (IBM PC/AT) was used together with an interface from Metrabyte and a signal conditioning interface. The diagram in Fig.3 shows schematically how the elements of the apparatus are interconnect

HARDWARE

DASH16 METRABYTE BOARD

Metrabyte's DASH16 is a multi–function, high–speed. analog/digital/input/output expansion board for personal computers. DASH16 uses an industry–standard (AD674A) 12 bit successive approximation converter with a 25fs conversion time. The channel input configuration is switch–selectable on the board providing a choice between 16 single-ended channels or 8 differential channels. Throughput using DMA (Direct Memory Access) channels can be up to 50,000 conversions/sec. Real time triggering of A/D assures perfect synchronization in all data acquisition not affected by other computer operations. These capabilities are essential for high-speed equipment control in a DOS environment where multi-tasking is not possible.

DASH16 has two channels of multiplying 12 bit D/A output. These converters are double-buffered to provide instantaneous update, thus allowing simultaneous control of the two servorams.

SIGNAL CONDITIONING INTERFACE

To interface servovalves, strain gauges, and LVDTs to the Metrabyte board, specialized electronic equipment was necessary. The equipment consists of two devices: first a board (Data Acquisition Interface Card) which occupies one of the slots of the IBM PC/AT, providing software-selectable amplification gains for eight channels and software-selectable analog hardware offset for four of the eight channels, second an A/C Signal Conditioning Unit which provides the excitation for the load cells and/or the LVDTs in eight channels. It also transforms the voltage output of the two D/A channels into currents, varying between +/- 15mA, to drive the servovalves.

SOFTWARE

The software is written in compiled BASIC version 3.0. However, when higher speeds are required ASSEMBLER subroutine calls are made from inside the BASIC program addressing specific components and/or requesting specific tasks.

Most of the interface with the Metrabyte DASH16 board was accomplished using the software provided by the manufacturer. The standard driver subroutines remain unmodified, allowing continued compatibility with the programming manual. However, five additional specialized driver subroutines were developed to increase the flexibility of the DASH16 board. These new driver subroutines are designated as MODE 19, 20, 21, 22, and 23.

For a clear understanding of the purpose of each of the different programs, an outline of the various tasks is necessary. While data is being collected, two servorams must apply independent and synchronized sinusoidal forces, with frequencies of up to 30 Hz, to a specimen. For higher accuracy in test results, feedback closed control loops and high signal resolution must be used. In order for several types of tests to be performed sequentially on a single specimen, rapid adjustment of signal ranges and rapid change of feedback channels is necessary, since the hydraulic servorams must always be kept under feedback closed looped control. Consequently, this forces the adoption of software-programmable gains and software-programmable analog voltage offsets. Furthermore, one should realize that feedback closed loop control derives its accuracy from the fact that the desired signal (COMMAND) is compared with the signal obtained at the probe (FEEDBACK), and an error signal is sent to the servovalves so they may make any necessary adjustments.

When analog equipment was used in feedback closed loop controls, the time between reading the feedback signal, comparing it to a command value, sending out the error value, and rereading the feedback value was almost zero, i.e. the rate (number of closed loops/time) was almost infinite. However, when using digital equipment there is time involved in:

–Converting the feedback signal from analog to digital
–Computing the command value
–Determining the error signal
–Converting the error signal from digital to analog

The rate is, therefore, a finite number which must be kept sufficiently high to provide complete control of the servorams. To accomplish this,command waves may be generated prior to testing,provided they are either periodic, or of short duration. This frees the computer's Central Processing Unit (CPU) from performing wave generation during testing. If the data acquisition components responsible for feedback data input are CPU independent, taking advantage of Direct Memory Access (DMA) channels, then error signal evaluation and output is the only task which must be executed during testing.

The menu-driven programs (SETUP, WAVGEN, and DACTEST) were developed so that the above requirements could be fulfilled.

SETUP

This program prompts the user to input the proper conversion factors, amplification gains, and analog hardware offsets for each of the eight input channels in order to convert input voltage from the probes into engineering units. The set of all these values constitutes a configuration file (<name>.

SET). When a specific configuration is required, the program retrieves the file from the Hard Disk, addresses the Data Acquisition Interface Card, and sets the required amplification gain and analog offset for each channel.

WAVGEN

The main function of this program is to create and to save an array containing the values of command waves to be used by DACTEST. The computer prompts the user with the name of the configuration file to be used with the command waves that it is about to generate. In order to define the waves the following information is requested:

- Feedback channel number for each of the servorams. Both of the servorams can be controlled with the feedback from any of the eight available channels. Stress or strain closed loop control is made possible simply by choosing the correct feedback channel.
- Frequency, amplitude, phase angle, wave form, and gain for each of the command waves. The software can perform feedback closed loop control at a rate of 5000 pts/sec/servoram. Therefore a 1 Hz sine wave would be defined by 5000 pts but a 50 Hz wave would be defined by 100 pts.
- Definition of data acquisition schedules. To accurately perform closed loop control, very high rates of data acquisition are necessary. However, for data analysis only a small number of conversions are required. Therefore, saving all the data points would be not only next to impossible but inefficient. The data acquisition schedule serves to instruct the computer on which periods to save for later analysis.

All this information is then saved in a test type file (<name>. WAV). Several types of tests can be created and saved by several runs of WAVGEN .

DACTEST

A testing sequence can consist of a series of different or identical tests, or even a single test, defined by the name of the file created by WAVGEN. For each test within the sequence the program requests a name for the file where the resultant data is to be stored. Since servorams must be kept constantly under control, and since opening a file requires a relatively long time, a macrofile where the data from all the tests will be stored must be opened before beginning each testing sequence.

All testing is actually controlled by MODE 20. The input for this subroutine is the output of the program WAVGEN; it contains all information pertinent to testing.

Every 1/5000 sec, DASH16, from its DMA controllers, updates 16 computer memory bytes (two for each input channel) with a digital value corresponding to the voltage present in each of the eight channels. Essentially, this allows values present in the probes to be place directly in memory (with only a 1/5000 sec delay) without spending CPU time. MODE 20 compares the values present in the bytes, representative of those in the feedback channels, with the corresponding values in the command waves. Control of the servorams is maintained by sending the difference between the feedback values and the command values, multiplied by a constant (GAIN) to the D/A channels of the DASH16 board. GAIN depends on the stiffness of the specimen, the oil pressure, the stiffness of the frame, and the frequency and amplitude of the loads. This constant must be determined experimentally in each situation.

Counters keep track of the number of periods tested, and when they equal a period number, pre-defined in the data acquisition schedule, all the values present in the memory bytes during one period are read and stored in another of the memory segments. This data is retrieved later and stored on the Hard Disk.

Exiting from this subroutine occurs only when each test is completed. At this stage the servoram would be completely out of control, if not for the sequential calls, placed between lines of BASIC, of MODE 22 . This mode was developed to allow complete control while the program performs other tasks such as data storage or input/output operations. Using MODE 22, manual control of the servorams is also possible. By this it is understood that by pushing the directional keys on the keyboard, command values for the servorams may be altered, forcing a change in their positions.

After the completion of each individual test, data is retrieved from RAM and stored on the hard disk (for each test 64Kb of data can be collected). This is accomplished while maintaining control of the servorams.

DATA ANALYSIS

Data analysis, for the most part, involves time-consuming, tedious, and repetitive tasks before reaching a level where engineering judgment can take place. Two post-processor programs (OPTMZER and ANALYSR) were developed to convert data obtained during a testing sequence into material properties.

OPTMZER

Data produced by DACTEST during a testing sequence is stored in a macrofile. OPTMZER reads the macrofile and creates a number of files (<name>.dat) equivalent to the number of tests in the testing sequence. For each of the tests, it also creates a file (<name>.OPT) containing the values of the amplitude, mean, and phase angle of the individual sinusoidal waves from each channel and in each period. These values are statistically obtained by the method of least squares. The information is converted and stored in engineering units, taking into consideration the amplification factor and the analog offset used during testing. This program is only executed once for each testing sequence since individual files are created to store the information to be processed alter by ANALYSR.

ANALYSR

Programs for data analysis derive their productiveness from the specificity of their function. ANALYSR is specifically designed to interpret data from dynamic tests retrieved with DACTEST. The program's menu–driven construction makes it user–friendly. The graphics capabilities of this program are extended by software libraries from the Graphics Development Toolkit. This software package provides valuable assistance in creating, displaying, and printing even the most elaborate graphics.

AN EDUCATIONAL PROJECT – DIAMETRAL TESTING DEVICE

The use of microcomputers in the classroom has already gained wide acceptance as a valuable tool for teaching and learning. Introducing them into a laboratory environment, where students may use them to perform experiments which are a routine part of class assignments, should prove invaluable. In the laboratory, microcomputers can execute a multitude of tasks and functions, ranging from data acquisition and equipment control to the visualization of the concepts or processes being tested. Furthermore, by completing tedious or repetitive tasks in a minimum amount of time, they help in creating an exploratory, mind stimulating environment. These capabilities are currently being exploited on the UC Berkeley campus, where students have been exposed to state–of–the–art developments in data acquisition and equipment control. An existing Diametral Testing Device, used to determine the resilient modulus of asphalt-concrete mixtures, was adapted and interfaced to an IBM PC/AT (see Fig. 4).

DATA ACQUISITION, EQUIPMENT CONTROL AND DATA PROCESSING

The software for the Diametral Testing Device is similar to that used in the Dynamic Loading System; however, it is designed to fulfill three significantly different requirements. Firstly, the programs are as user-friendly as possible in order to simplify use by students. Secondly, the data is displayed, processed, and graphed in as close to real time as possible to facilitate the students' immediate visualization of the test. Finally, the open loop control needed to drive the solenoid valve is significantly less demanding on the software than the feedback closed loop control required by hydraulic servovalves.

RESMOD is a menu-driven program designed with these considerations in mind. With this program, three options, each corresponding to one part of the testing sequence, are available. The options are as follows:

Setting the load – The solenoid valve opens, pressurizing the piston. Readings from the load cell, displayed in engineering units, allow corrective adjustments of the load to be made using the pressure knob.

Placement of LVDTs – The LVDTs are positioned on the yoke, and the screen displays their readings in engineering units.

Testing – The program prompts the user to input the parameters defining the tests, such as the number of cycles, time of load on (usually 0.3 sec), time of load off (usually 2.7 sec), and specimen thickness.

To allow nearly simultaneous data acquisition (from the load cells and LVDTs), data processing (computation of the resilient modulus), and data graphing (display of the raw data and a diagram of the deformed specimen), the DMA capabilities of the Metrabyte DASH16 Board were again employed.

Data from each load cell is alternately stored in one of two memory segments. While the DMA channel is filling one memory segment with data from one period, the CPU is retrieving data from the previous cycle, stored in the other memory segment. Displaying the data is achieved using call functions from the Graphics Development Toolkit. After all the cycles have been completed, the values of the resilient modulus obtained from each cycle are displayed and printed if necessary.

Since the loading times and loads are easily changed and the data from the experiments is processed and graphed in almost real time, students can readily perceive the rheological behavior of asphalt mixtures.

CONCLUSION

The development of the Dynamic Loading System, making possible the determination of dynamic properties of pavement materials, is an example which fully illustrates microcomputer capabilities in a research environment. Beyond the economic advantage of replacing expensive equipment, i.e. strip chart recorders, function generators, and closed–loop controllers, microcomputers present: 1) the added advantage of eliminating the need for time–consuming data manipulation; and 2) the ability to control several processes involving different pieces of equipment simultaneously.

In an educational enivronment microcomputers have proven to be a valuable teaching tool, since they combine the above advantages with extensive graphics capabilities. Through proper programming the students' visualization of the concepts or processes involved may be increased; moreover, the tedium of repetitive calculations may be reduced thus leading to a more stimulating classroom environment.

ACKNOWLEDGEMENTS

Partial funding for this project was provided by the Transportation Laboratory of the State of California Department of Transportation, International Business Machines (IBM), and the Institute of Transportation Studies (ITS).The authors wish to express their appreciation for the contribution provided by Manuel Bronstein during the initial development of the ASSEMBLER code and by Paul Gross for the design and construction of all electronic interfaces.

REFERENCES

Lysmer, J.L., C.L. Monismith, J.B. Sousa and S.S. Chen, Pavement Response to Dynamic Loads. Report to the Transportation Laboratory, State of California Department of Transportation, University of California, Berkeley, October, 1985.

Schmidt, R.J., " A Practical Method for Measuring the Resilient Modulus of Asphalt-Treated Mixes," Highway Research Record No. 404, HRB, 1972.

Sitar, N. and Y.D. Wang, "Cyclic Test Control And Data Acquisition Using A Microcomputer," Proceedings, 2nd National Conference on Microcomputers in Civil Engineering, Orlando, Florida, 1984, pp.6-8.

Sousa, J.B. (1986): "Dynamic Properties of Pavement Materials." Ph.D thesis in preparation. University of California, Berkeley.

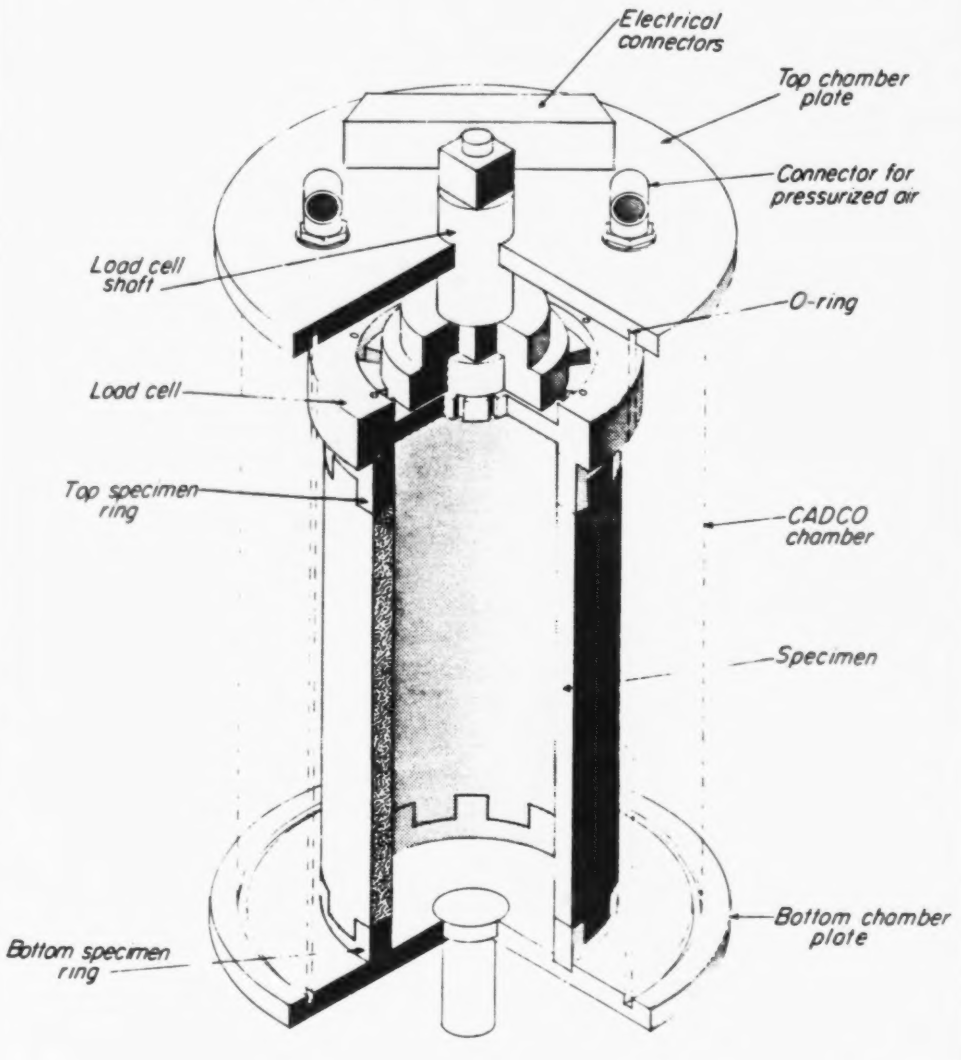


Figure 1. Axonometric view of pressure chamber, specimen and load cell.

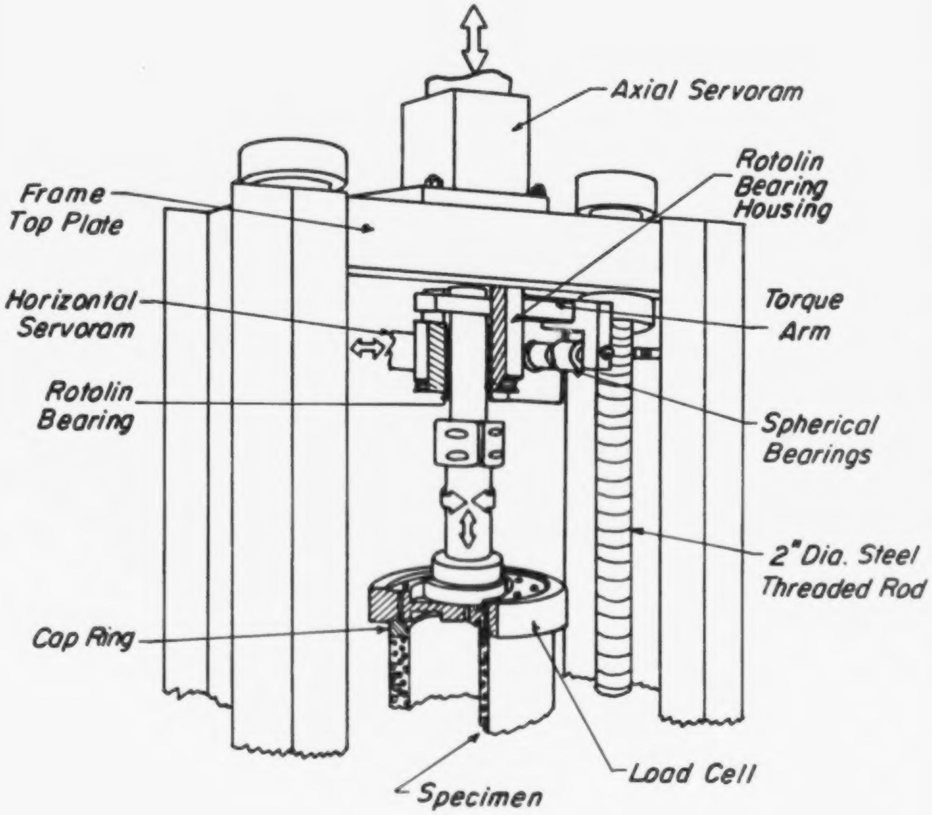


Figure 2. Loading System.

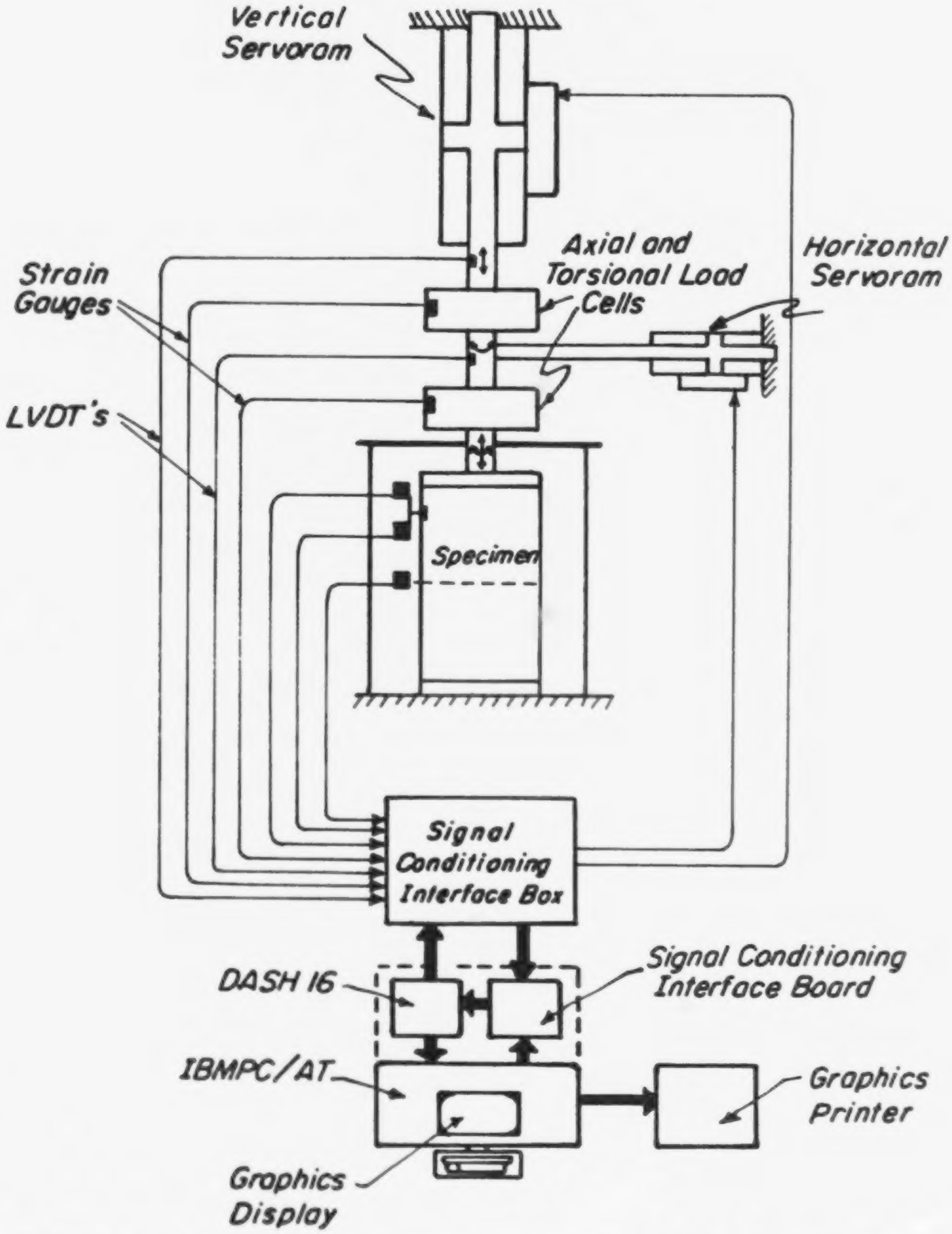


Figure 3. Data Acquisition and control scheme – DYNAMIC LOADING SYSTEM.

520

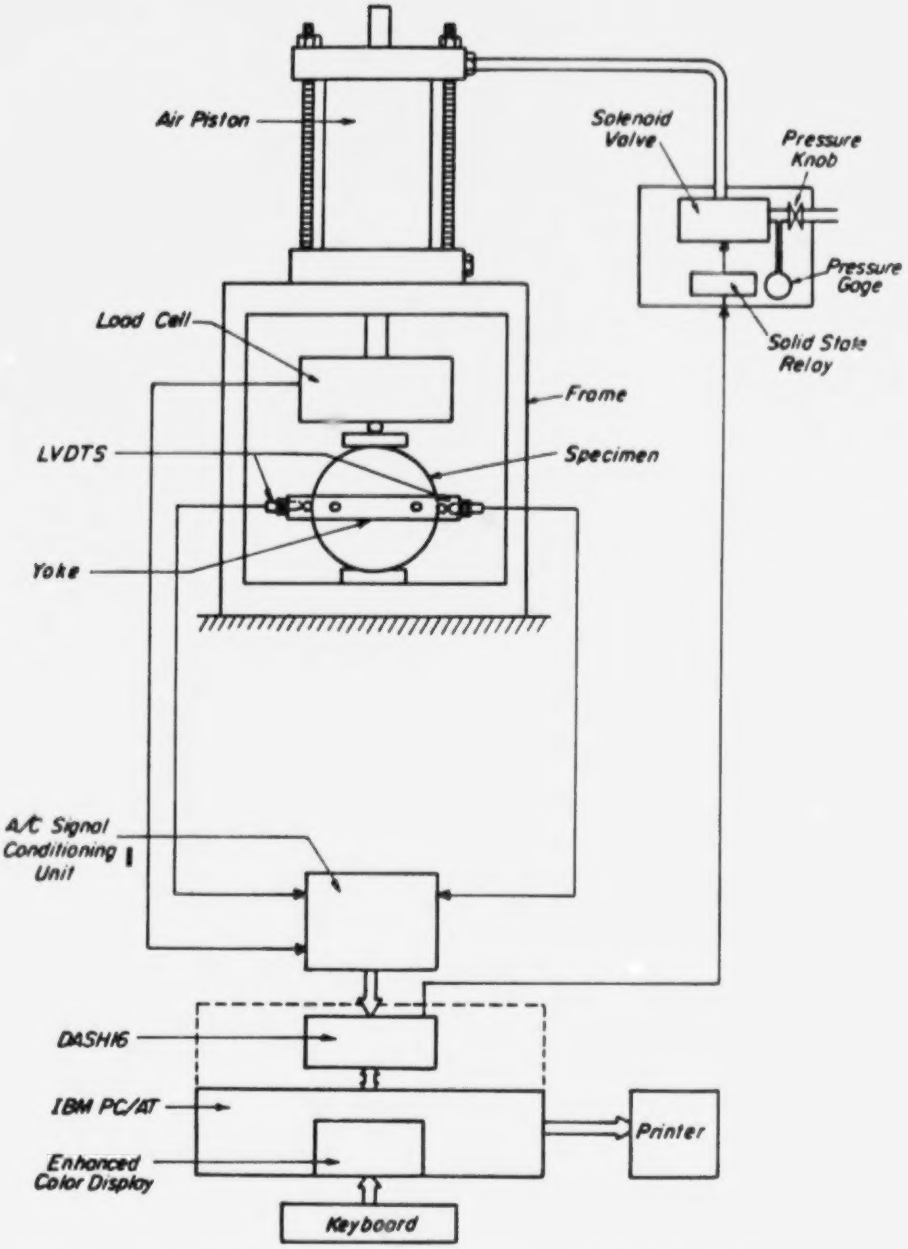


Figure 4. Data Acquisition and control scheme – DIAMETRAL TEST.

COMPUTER–AIDED DESIGN OF MICROWAVE INTEGRATED CIRCUITS

Presenter: **David Rutledge** *California Institute of Technology*

Coauthors: **David Rutledge** *California Institute of Technology*
Richard Compton *California Institute of Technology*
Wyman Williams *California Institute of Technology*

ABSTRACT

Caltech has a one–quarter class for seniors and graduate students in the computer–aided design of microwave integrated circuits. In the class, the students make a variety of microstrip circuits and devices, including a branch–line coupler, matching networks, a patch antenna, a monopulse array, a mixer, and a low–noise amplifier. The students design the microwave circuits on an IBM PC with a Turbo Pascal program, Puff, that can lay out and analyze the circuits, and make the photographic artwork on a dot–matrix printer. The circuits are tested by a network analyzer controlled by an IBM PC, with a Turbo Pascal program, Elf, that leads the students through a calibration run, makes error corrections, and plots the data.

MICROWAVE INTEGRATED CIRCUITS

Microwave circuits are different from low–frequency circuits because the circuit size is comparable to a wavelength. A microwave engineer usually thinks not in terms of voltages and currents, as in a low–frequency design, but rather in terms of incident, reflected, and transmitted waves. This approach is particularly appropriate for the frequency range from 1 GHz (10 9 Hz) to 1 THz (10 12 Hz), where the wavelengths are between 30 cm and 0.3 mm. The most important microwave frequencies are from 1 GHz to 10 GHz, because these wavelengths propagate extremely well through rain and snow. Microwave circuits are central components in radar, aircraft and ship navigation, ground relays, satellite communication, and cooking and drying.

Traditional microwave hardware looks more like plumbing than electronics. The signals propagate down hollow rectangular pipes, and there are bends and T junctions. The signals are generated and amplified by vacuum tubes. In the last ten years, however, a new microwave integrated–circuit technology has evolved, based on microstrip transmission lines (Fig. 1), and new fast gallium–arsenide transistors and Schottky diodes. Microstrip transmission lines resemble ordinary printed–circuit boards, and are etched and processed in essentially the same way. However, the return current is carried by a ground plane on the back of the circuit board, and strict attention is paid to the dielectric constant of the substrate, because the circuit elements are formed from different sections of transmission lines of varying length and width. These substrates and devices are much cheaper than the machined hollow metal guides. The microstrip substrates are typically $1 per square inch, and the transistors and diodes cost about $10. Metal waveguide components and microwave tubes, on the other hand, typically start at prices ten times as high. In addition, the fabrication process for integrated circuits, which is based on photolithography, is much quicker than machining the old guides.

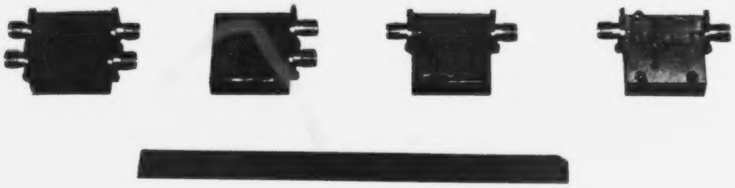


Figure 1.
Microstrip circuits made in the Caltech microwave integrated – circuits class. From left to right, a coupler, antenna, amplifier, and oscillator.

These changes have created the opportunity for a new kind of microwave integrated–circuits class, where the students lay out and analyze the circuits on a personal computer, and make the photographic artwork. They can then make the circuit and test it. Last fall we started a one–quarter, combined lecture and laboratory class with 12 students. The students listen to two fifty–minute lectures each week on designing the circuits. They then designed the circuits for homework, and made them in a three hour lab. They each made six circuits: a directional coupler, and patch antenna, an impedance matching circuit, a monopulse antenna array, a mixer, and an amplifier. For the last three weeks of the term, the students worked on individual projects, which ranged from designing and building oscillators and arrays of antennas to writing computer–aided design software.

DESIGN

There are several problems with the available software for designing microwave integrated circuits on personal computers. There is no one industry standard, but there are several programs written by specialized companies that typically cost about $10,000. The price for a university is usually a small fraction of this, but we have found that the restrictions of copy–protection and hardware keys to be a major stumbling block for a campus with a large number of personal computers. The programs are written for industrial users that frequently have a very different focus from the students. The complexity of the programs is enormously greater than necessary, and this makes the programs slow and hard to use. To give an example, an industrial engineer may need to know the minute difference in performance that using different metals in the circuit makes, but in a class, having a choice between several different metals is distracting.

In addition, the source code and the design algorithms are proprietary, and this creates several difficulties. Because the source code is unavailable, a student cannot see what the program does or how it does it, and the program cannot be modified. It gives precisely the wrong idea about a design program, in that it sets the computer up as an oracle that cannot be questioned. In reality, the analysis is approximate, with limited validity for a limited range of variables. There are errors in the physical approximations, the mathematical algorithms, the construction of the circuits, and the testing. The student must appreciate how big the different errors are, and he cannot do it without seeing the source code.

Because of this, we have been developing a program called Puff, after the dragon in the folk song. Puff is written in Turbo Pascal for an IBM PC with the Enhanced Graphics Adapter and the source code is 100 kilobytes long. Pascal is particularly suitable for writing source code that another person will read. The advantage of using the Enhanced Graphics Adapter is that the circuit layout and analysis plots can be on the screen at the same time. The user draws a microstrip circuit on the screen with the cursor keys (Fig. 2). The frequency response for a particular circuit can be plotted at any time. Puff analyzes the layout, and makes a Smith chart plot, which is a polar plot of the various reflected and transmitted waves in the system as a function of frequency. The analysis approach is to break the circuit down into small components with a known frequency response. Then the program goes through the list of components, combining them in pairs until the entire circuit is assembled. Solving the circuit at a single frequency is equivalent to solving a rather sparse system of 20 linear equations with 50 complex coefficients. A typical frequency plot requires solving 20 different sets of these equations. In addition, rectangular plots can be made of the magnitude and phase, and the time-domain response can also be plotted by taking a fast-Fourier transform of the frequency response. Analyzing and plotting for the students' circuits typically requires 20 seconds on a PC and 10 seconds on an AT.

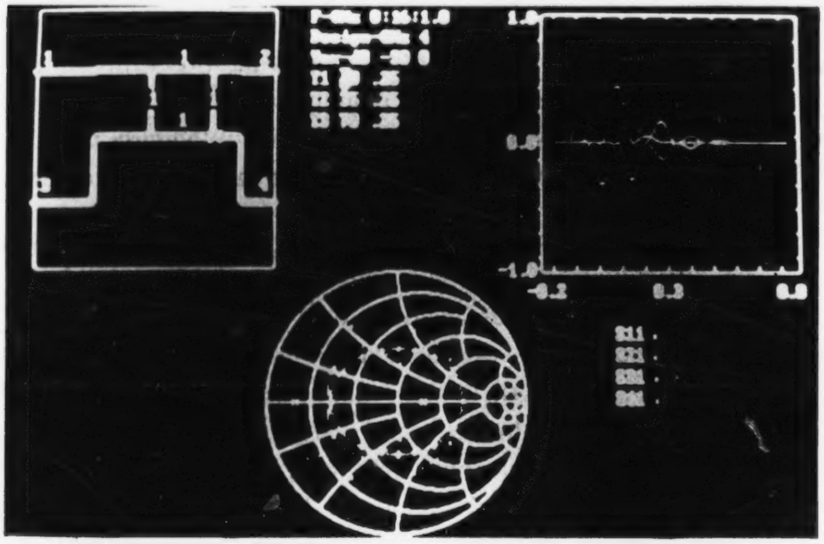


Figure 2.
Puff Display on an IBM AT with an Enhanced Graphics Adapter. The circuit layout is in the upper left corner, the polar plot (Smith chart) is on the bottom, and the rectangular plot is on the right.

Puff also makes the photographic artwork directly from the screen layout on an IBM graphics printer (Fig. 3a). Two passes are made over each line to achieve a resolution of about 100 dots per inch. The artwork is then photographically reduced 5:1 onto a glass plate (Fig. 3b), so that the final feature accuracy is about two mils. For comparison, the typical line widths vary from 10 mils to 100 mils. The students use the glass plate to expose photoresist on the microstrip substrate. The photoresist patterns are developed and the excess copper etched away. The circuits are mounted onto a brass block, and coaxial connectors are added. The devices are soldered on, and the circuit is complete (Fig. 1). The entire fabrication can be done in a single three-hour lab period.

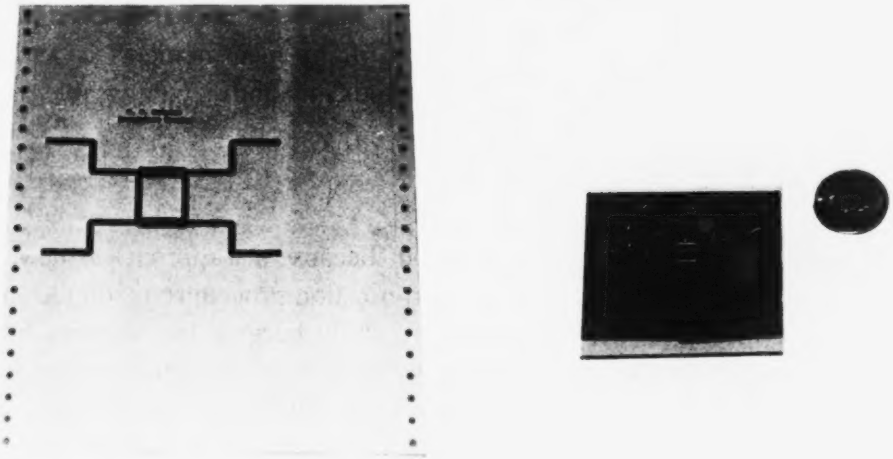


Figure 3.
Photographic artwork for the IBM graphics printer (a), and reduced 5:1 onto a glass plate (b).

TESTING

Testing presents different problems. Excellent equipment is available, but it is extremely expensive. The current industry standard is the Hewlett-Packard 8510, and, fully equipped, it costs $150,000. Hewlett-Packard has donated single analyzers to some universities with large microwave programs, but universities with smaller programs do not have access to them. Even the universities that receive the instruments have to figure out how a large number of students can learn from a single instrument. The reason the equipment is expensive is that microwave measurements are rather complicated. Both the hardware and software are sophisticated. It is generally not possible to measure a signal where one would like because one cannot put a probe there. One has to run lines to the side of the board and add connectors and coaxial cables. One then applies signals to the coaxial lines and measures the reflected and transmitted waves. One then refers these waves to the appropriate position in the circuit. Figuratively, it is like trying to figure out what a mountain looks like by shouting and listening for the echoes.

To solve this problem we have written a computer program, Elf, for the IBM PC. Elf is also in Turbo Pascal, and the source code is 100 kilobytes long. Elf runs an earlier Hewlett–Packard network analyzer system, the 8410, through a Data–Translation A–D and D–A board and a Capitol Equipment IEEE–488 Bus. The 8410 system "only" costs $50,000. Elf guides the students through a series of calibration steps, measures the data, and makes a series of complex error corrections on the data. The accuracy of the measurements is about a factor of ten better than normal 8410 measurements, and is comparable to the specifications for the newer 8510 analyzer.

CONCLUSIONS

Many things in the class need to be improved. Because the approach is new, the class will need a book. At three hours, the fabrication procedure is still too long. Supplies were expensive, $100 per student, partly because the students broke many of the delicate connectors. The cost would have been higher, except for a generous donation of microstrip substrates from the Rogers corporation. The students spent a great deal of time waiting for their turn to use the network analyzer. Only a few of the final projects were well done. The students tended to be too ambitious, and our supervision was too loose. Many were just worn out at the end of the term from other classes. On the other hand, the students were enthusiastic and the circuits worked well. Some of the amplifiers had noise figures of less than 2 dB at 4 GHz. A similar class has now been started at UCLA under Professor Neville Luhmann. The students that took the class now have a lot of experience in building and testing these circuits.

COMPUTER GRAPHICS FLOW VISUALIZATION

Presenter: W. L. Olbricht *Cornell University*

Coauthors: W. L. Olbricht *Cornell University*
M. U. Nollert *Cornell University*
G. P. Moromisato *Cornell University*

An interactive computer graphics program has been developed to help introduce basic concepts of fluid motion through visualization of two-dimensional linear flows. The kinematics of flows specified by the user are displayed in color animation. The time-dependent motion of an axisymmetric rigid particle immersed in the flow also can be displayed, and the effects of bulk kinematics on the orientation of the particle are readily exposed.

1. INTRODUCTION

A challenge in teaching fluid mechanics at the undergraduate level is to make apparent the relationships between the formal mathematical description of fluid flow and the physical motions that fluids exhibit. Students often study in detail a small number of selected "prototype" flows that are presented mainly to illustrate different methods of solving the equations that govern fluid dynamics. However, in many flow problems that are likely to be encountered in practice, exact solutions of these governing equations either are not available or are extremely difficult to obtain. In these cases, a qualitative understanding of the possible microscale motions of the fluid can be a significant advantage in analysis and design. For example, the mixing of fluids is an important operation that is common to many processes. The efficiency of fluid mixing apparently depends on the details of complicated flow patterns that cannot be calculated exactly. Still, certain kinds of motion are known to enhance mixing, and these motions can be identified qualitatively, and even classified, according to relatively simple kinematic considerations.

It may be useful then to have available for students studying fluid dynamics for the first time a tool to compare the kinematics of various flows that are members of certain well-defined classes of motions. This tool could be used to display the kinematics of flows specified in terms of mathematical parameters that are introduced formally in typical introductory fluid mechanics courses and texts. This formal description involves the use of vectors and tensors, which can complicate presentation of the material. The graphics tool for displaying fluid kinematics could help students relate the mathematical form of tensorial parameters to actual physical motions. The major advantage of a computer–based tool is that tedious arithmetic computation required to construct representations of flows could be done automatically, allowing students time to study many specific examples, and to deduce certain principles that otherwise would have to be "announced".

The same tool that calculates and displays flow kinematics can be readily adapted to study the motion of particles that are immersed in the flow. The behavior of a particle immersed in a flow with known "bulk" kinematics is a fundamental problem in the hydrodynamics of suspensions, which is of considerable practical interest in the processing of multiphase materials, such as fiber–filled composites, immiscible emulsions, and polymer solutions. The most important principles to illustrate in this case are that the motion of a suspended particle does not necessarily coincide with that of the surrounding fluid, and that certain flows can cause particles to align in preferred spatial orientations. The degree of alignment of particles can strongly influence the material properties of the final product.

The motion of suspended particles in simple, well–defined flows can be quite complicated and is generally time–dependent, even when the bulk kinematic field is steady. As a result, this subject is usually reserved for graduate–level courses that can present detailed solutions of the problem with appropriate mathematical rigor. Again, though, it would be useful to have a simple tool that beginning students could use to gain some insight into basic principles of particle motions and suspension mechanics.

This paper describes an interactive computer graphics program to help introduce basic concepts of fluid motion through flow visualization for the broad class of linear two–dimensional flows. The program is supplied to students enrolled in an undergraduate–level course in fluid dynamics. The theoretical background required to use the program effectively is presented to students in class, and the necessary concepts are summarized in section 2. The computer algorithm and graphics display is described in section 3, and investigations that student–users can undertake are suggested in section 4.

2. THEORETICAL BACKGROUND

Fluid motion for the class of linear, homogeneous, two–dimensional, steady flows is specified as follows:

$$\underline{v} = L \cdot \underline{x} \qquad (1)$$

where $\underline{v}$ is the velocity vector at the location $\underline{x}$, and L is a matrix containing velocity gradients as it components and is identified with the transpose of the velocity gradient tensor. [Calculations are simplified by employing Cartesian coordinates so that expressions involving tensors can be calculated using matrix algebra.] Thus, the components of L are $L_{ij} = av_i/ax_j$, where i,j = 1,2. For incompressible fluids, continuity requires that $L_{11} + L_{22} = 0$. It is clear that the kinematics of the flow are completely determined by the components of L and that the fluid velocity at any location can be directly computed from (1). Alternatively, the trajectory of fluid elements that pass through a given point in the flow field can be calculated. If $\underline{v}$ in (1) is replaced by $d\underline{x}/dt$, then (1) can be regarded as a set of differential equations that show how the location of a fluid element with a certain initial position changes in time. Then the map of fluid trajectories for two–dimensional flows is equivalent to the phase portrait of the system (1). In fact, the trajectories of fluid elements form a subset of trajectories of (1) with $L_{11} = -L_{22}$.

The qualitative form of the fluid trajectories is governed by the eigenvalues of L, which are easily calculated for two–dimensional flows as:

$$\lambda_i = \pm \sqrt{L^2_{11} + L_{12} L_{21}} \qquad . (2)$$

The associated eigenvectors $\underline{e}_i$ have the following components:

$$\underline{e}_i = \left[1, \frac{\pm \sqrt{L^2_{11} + L_{21} L_{12}} - L_{11}}{L_{12}} \right] \qquad . (3)$$

It is often useful to split L into a symmetric tensor E and an anti–symmetric tensor Ω:

$$E = (L + L^T)/2$$
$$\Omega = (L - L^T)/2 \qquad (4)$$

where L^T denotes the transpose of L. The rate of deformation tensor E describes a pure straining motion, and the vorticity tensor Ω corresponds to a rigid body rotation. The components of the vorticity tensor Ω and the associated vorticity vector $\underline{\omega}$ are related according to $\underline{\omega} = \epsilon:\Omega$, where ϵ is the alternating tensor with components ϵ_{ijk} The separate contributions of E and Ω to the overall motion can be examined as described in section 4.

A freely suspended rigid particle immersed in a flow that has kinematics specified by the tensor L will translate and rotate in the flow in order to make the net force and torque on the particle zero. This allows determination of the particle trajectory, and the problem has been solved for certain classes of particle shapes, including spheroids of revolution. In this case, the particle translates with a velocity equal to the undisturbed velocity of the fluid, i.e. with no particle present, at the location occupied by the center of mass of the particle. The rotation of the particle can be described in terms of a unit vector $\underline{r}$ embedded along the symmetry axis of the spheroid, as shown in Figure 1. The result is given in terms of the time rate of change of the vector $\underline{r}$ as follows [1]:

$$d\underline{r}/dt = \Omega \cdot \underline{r} + G\,[\,E \cdot \underline{r} - (\underline{r} \cdot E \cdot \underline{r})\,\underline{r}\,] \,. \qquad (5)$$

The parameter G is a constant that specifies the shape of the spheroid and is related to the aspect ratio of the particle r according to $G = (r^2 - 1)/(r^2 + 1)$. Thus, G varies over the interval $(-1, 1)$. Prolate spheroids have values of $G > 0$, and oblate spheroids have $G < 0$, with the special case $G = 0$ corresponding to a spherical particle. The case of very thin rigid rods, which is of special technological importance, is an asymptotic limit given by $G \rightarrow 1$. Equation (5) describes the rotation of the spheroid correctly whether or not the initial orientation of the spheroid is such that the vector $\underline{r}$ is contained in the plane of the flow, i.e. the x_1–x_2 plane in Figure 1.

3. PROGRAM DEVELOPMENT

In order to demonstrate the concepts outlined in sections 1 and 2, an interactive graphics program has been developed for the IBM PC-AT equipped with the Enhanced Graphics Adapter and an IBM Proprinter. The program allows the user to observe the kinematics of any linear two-dimensional flow and the dynamics of a particle suspended in such a flow. The graphics program can display fluid kinematics either as a "snapshot" of the instantaneous velocity field, as a "map" of fluid trajectories that pass through specific points, or in actual "real–time" animation. The motion of a suspended particle, which is inherently time–dependent even when the imposed velocity field does not vary in time, is displayed in animation. The graphics images are accompanied by a display of certain numerical data that allow quantitative calculations to be made by the user.

The flow to be examined is specified by the user by entering values for components of the tensor L. The computer then draws on the screen a number of "vectors" which indicate the direction and magnitude of the local fluid velocity at various points. A Proprinter output of the screen at this stage is shown in Figure 2a. The user can display the trajectory of a fluid element that passes through a point of interest by moving the cursor to that point and issuing a command to display the specific "pathline". Several such pathlines are illustrated in Figure 2b.

These displays are helpful in gaining a qualitative understanding of the "global" kinematics of the specified flow, but they are not very useful in observing the motion of fluid elements relative to other fluid elements, which is essential to understanding concepts such as stretching and orientation of "fluid lines". For this purpose the graphics program can display the motion of labelled fluid elements in real time. The user can "mark" a fluid element by positioning the cursor at a specific point and issuing a command to place a marker on that "piece" of fluid (cf. Figure 2c). When the user issues a command to start motion, the marker moves identically with the fluid velocity at the location of the marker (cf. Figure 2d). By labelling several fluid elements, the user can observe the orientation of adjacent fluid elements (which define a "line element" of fluid) and the rate of separation between the elements. These will depend strongly on the specific form of the tensor L.

The animation display is also used to observe the motion of a spheriod placed in the flow. The user specifies the initial location and orientation of the particle by placing the cursor at the endpoints of the particle. The geometry of the particle is specified by entering the particle aspect ratio r_e . The program displays the axis of revolution of the particle as shown in Figure 2c. When the start command is issued, the particle moves according to the dynamical equation (5). For cases when the particle is not totally contained in the two–dimensional plane of the flow, the display still shows correctly the projection of the vector $\underline{r}$ in that plane.

Figure 2 shows also the current user–specified elements of the tensor L in the upper right corner of the screen. The bottom line indicates the active menu. The following choices, selected by a single letter, are available to the user:

<space> – Draws the fluid trajectory that passes through he point where the cursor is positioned.

<n>ew Eq – Enter new coefficients for the velocity gradient matrix L.

<e>lements – Allows the user to "mark" points in the fluid or to place a particle with any orientation and aspect ratio. As elements are placed on the screen, their coordinates are displayed on the right hand side of the screen.

<g>o – Starts the animated display of the motion of the labelled fluid element(s) and/or particle(s).

<l>ocate – Displays the coordinates of the cursor location.

<z>oom – Enlarges a portion of the screen.

<P>rint – Prints the screen on an IBM Proprinter.

<Q>uit – Ends the program and returns to DOS.

Several menu options have sub–menus, and current options are always displayed on the screen.

The motion of the fluid and the rotation of a suspended particle are governed by the sets of differential equations (1) and (5), respectively. These equations are solved numerically using a fourth–order Runge–Kutta integration scheme. Because fluid trajectories are constructed numerically, the program could be readily adapted to display fluid motion for other flows, provided expressions are available for the velocity components as functions of position. Thus, in homogeneous flows, such as flow around a cylinder or flow through a conduit could be studied using the present code by changing only the program lines that specify the fluid velocity. The program must then be recompiled and linked, but all of the other parts of the algorithm, including graphics output, will work in the way described above.

4. PROGRAM IMPLEMENTATION

The most important feature of the program described in section 3 is that students are able to visualize the kinematics of many linear two-dimensional flows in a very short period of time and without extensive computations. Judicious choices for the flows to be examined enables users to expose, by a series of examples, certain principles that relate flow kinematics to the mathematical parameters outlined in section 2. In this section, several investigations that can be undertaken by users, and "conclusions" that can be drawn are discussed.

Examining the kinematics of several flows should be sufficient to indicate that the fluid pathlines can be either open or closed, depending on the exact choice for the tensor L. In every case, the origin is a "stagnation point" of the flow. When the fluid pathlines are closed, labelled fluid elements exhibit a periodic motion that follows elliptical paths about the origin. The maximum separation between fluid elements that are "initially" in close proximity is bounded. On the other hand, when streamlines are open, the motion of fluid elements is not periodic, and the separation distance between fluid elements initially in proximity increases without bound. Thus, the stagnation point is either a "center" for closed trajectories or a "saddle point" for open trajectories. Since fluid mixing is enhanced by significant stretching of line elements of fluid, flows that have closed pathlines are likely to be less efficient in mixing than flows that have open streamlines.

By correlating the eigenvalues of the tensor L with the qualitative picture of fluid pathlines for a series of flows, it should become apparent that the flows with open streamlines correspond to cases where the eigenvalues of L are real, and flows with closed streamlines have eigenvalues of L that are purely imaginary. When the eigenvectors of the flows are calculated from (3), they will be seen to coincide with the axes of the elliptical paths for closed-streamline cases, or to the asymptotes of the fluid trajectories for open-streamline cases.

The decomposition of the tensor L into a symmetric E and an anti-symmetric tensor Ω according to (4) is often displayed in introductory textbooks for the single case of simple shear flow. The tensor E specifies a pure-straining motion while the tensor Ω describes a rigid body rotation. This can be easily generalized to any choice for L by displaying the kinematic fields associated with E and Ω calculated from (4). The rate of stretching of a fluid element, and the angular velocity of a fluid element, which are easily calculated from screen displays, can be correlated with the magnitudes of E and Ω, respectively. In the special case of simple shear flow, these magnitudes are equal.

After studying several flows, users of the program can be encouraged to examine cases with the following components for the tensor L:

$$L = \begin{bmatrix} 0 & 1 \\ \alpha & 0 \end{bmatrix} \tag{6}$$

for several values of α. It will be apparent that by varying the single parameter a over the range $-1 \leq a \leq 1$, the entire range of possible flow fields is spanned, from a pure-straining motion (hyperbolic extension) for $\alpha = 1$, through simple shear for $\alpha = 0$, to pure rotation for $\alpha = -1$.

When the dynamical problem of the rotation of a rigid spheroid of revolution is considered, the difference between the motion of a suspended particle and the kinematics of the undisturbed flow should be readily apparent. By "placing" a rod in the flow and labelling fluid elements that initially coincide with the endpoints of the particle, it will be seen that the orientation of the spheroid will differ from the orientation of a line element of fluid for all later times. Furthermore, certain flows will cause the particle to take up preferred orientations. In these cases, the spheroid rotates from its initial orientation specified by the user to an asymptotic orientation for large times. Other flows cause the particle to exhibit a periodic motion with an angular velocity that varies throughout the period of rotation. It should become apparent that flows that cause the alignment of line elements of fluid do not necessarily produce orientation of the suspended particle, and vice versa. The user should find that the geometry of the particle plays an important role in determining the motion of the particle. In fact, the particle tends toward a fixed orientation when the eigenvalues of the tensor formed by the sum Ω + GE are real; a line element of fluid tends toward a fixed orientation with the eigenvalues of L are real. For the special cases of purely rotational flow, i.e. E = 0, the particle moves identically with the fluid in rigid body rotation.

The case of simple shear flow is noteworthy because the particle trajectory corresponds to the well–known Jeffery orbits2. The period of the orbit will be seen to depend on the geometric factor G. The limiting case G ––> 1 yields the behavior of an infinitely thin rigid rod. In this case, the period of the Jeffery orbit tends to infinity, and the rod tends to "align" in the flow.

5. CONCLUSIONS

The presentation of introductory fluid mechanics can be enhanced by having a simple tool to construct representations of flow fields without extensive arithmetic computation. The computer graphics program developed in this project allows students to visualize a broad class of motions, to explore many examples of specific flows economically, and to correlate their observations of fluid motion with the formal mathematical parameters that describe fluid kinematics. An advanced application such as the motion of a suspended particle can be introduced in a first course in the context of the interactive graphics presentation.

6. REFERENCES

1 Bretherton, F. B. (1962) The motion of rigid particles in a shear flow at low Reynolds number. J. **Fluid Mech**., 14, 284.

2 Jeffery, G. B. (1922) The motion of ellipsoidal particles immersed in a viscous fluid. Proc. Roy. Soc., A102, 161.

FIGURE CAPTIONS

Figure 1:
The orientation of any spheroid of revolution is represented by the vector, r, embedded along the symmetry of the particle.

Figure 2:
a) The kinematics of the flow is represented by a vector field showing the direction and magnitude of the local fluid velocity.
b) The path lines of several fluid elements have been added.
c) A particle with an aspect ratio of eight has been added to the screen and four fluid elements have been "marked" with + signs.
d) The particle and "marked" fluid elements begin to align when the flow is started.

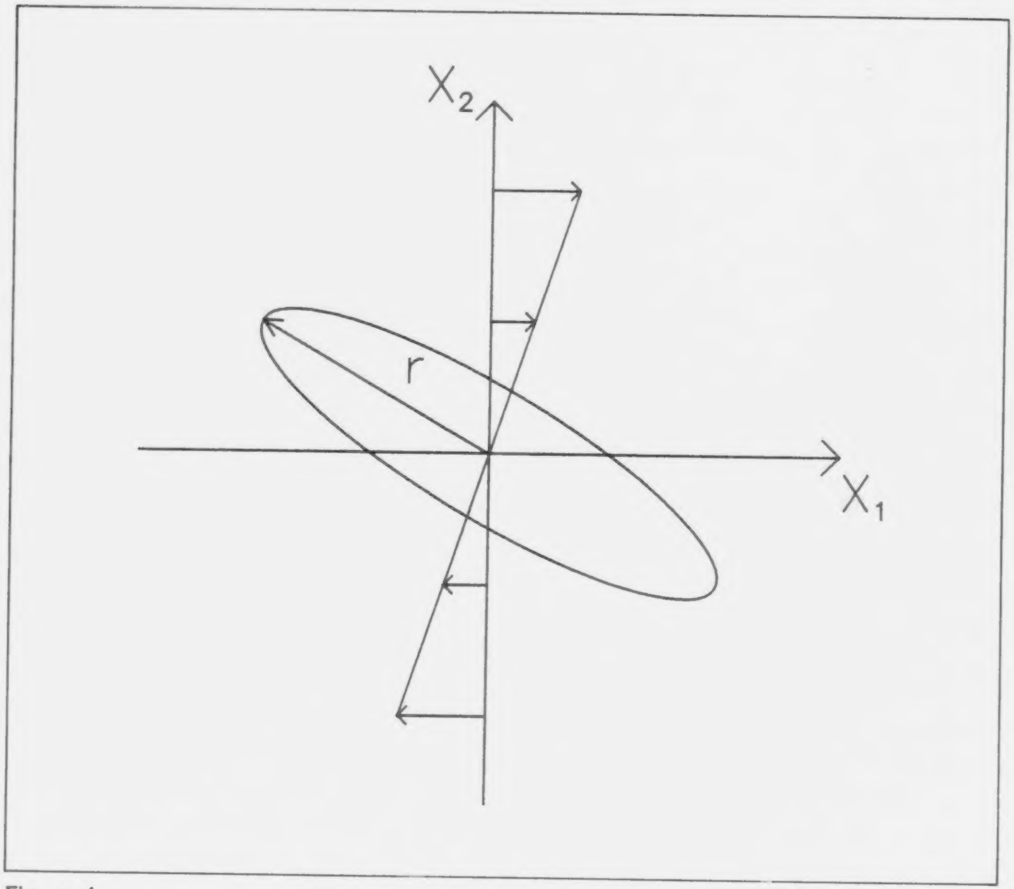
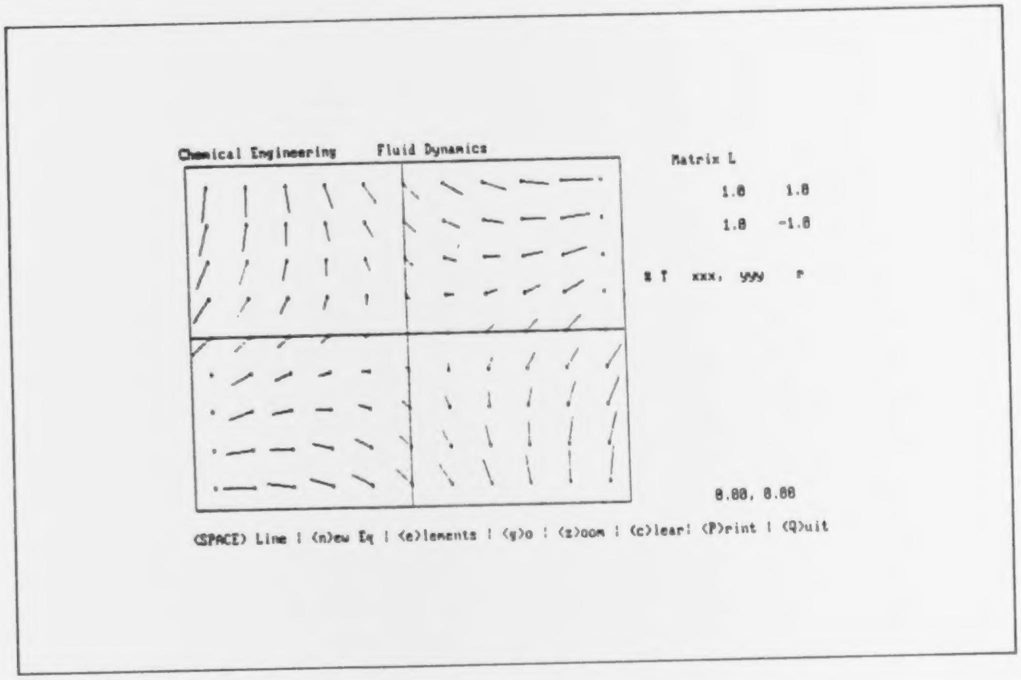
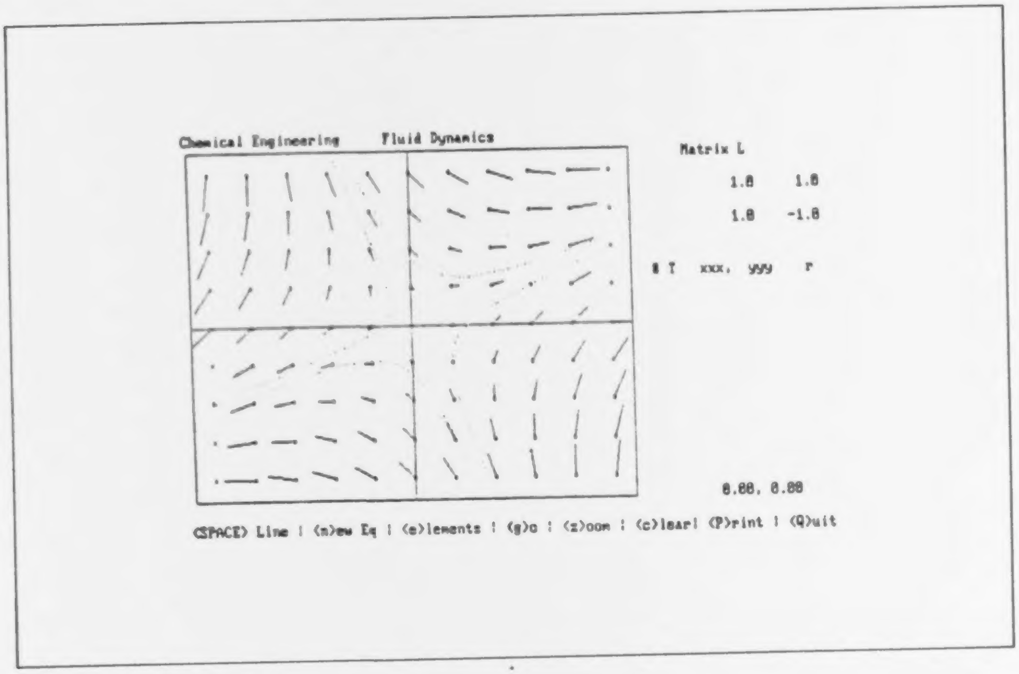


Figure 1.


Chemical Engineering
Fluid Dynamics
Matrix L
# T xxx, yyy r
0.00, 0.00
<SPACE> Line | <n>ew Eq | <e>lements | <g>o | <z>oom | <c>lear| <P>rint | <Q>uit

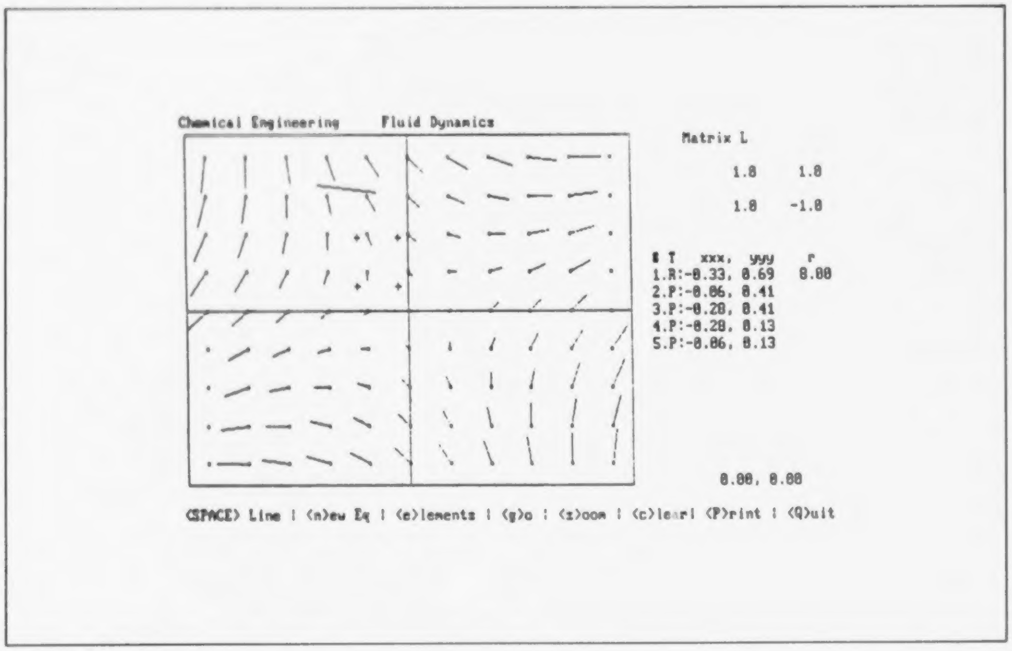
```
Matrix L
1.0   1.0
1.0  -1.0
```

Figure 2.a


Chemical Engineering
Fluid Dynamics
Matrix L
# T xxx, yyy r
0.00, 0.00
<SPACE> Line | <n>ew Eq | <e>lements | <g>o | <z>oom | <c>lear| <P>rint | <Q>uit

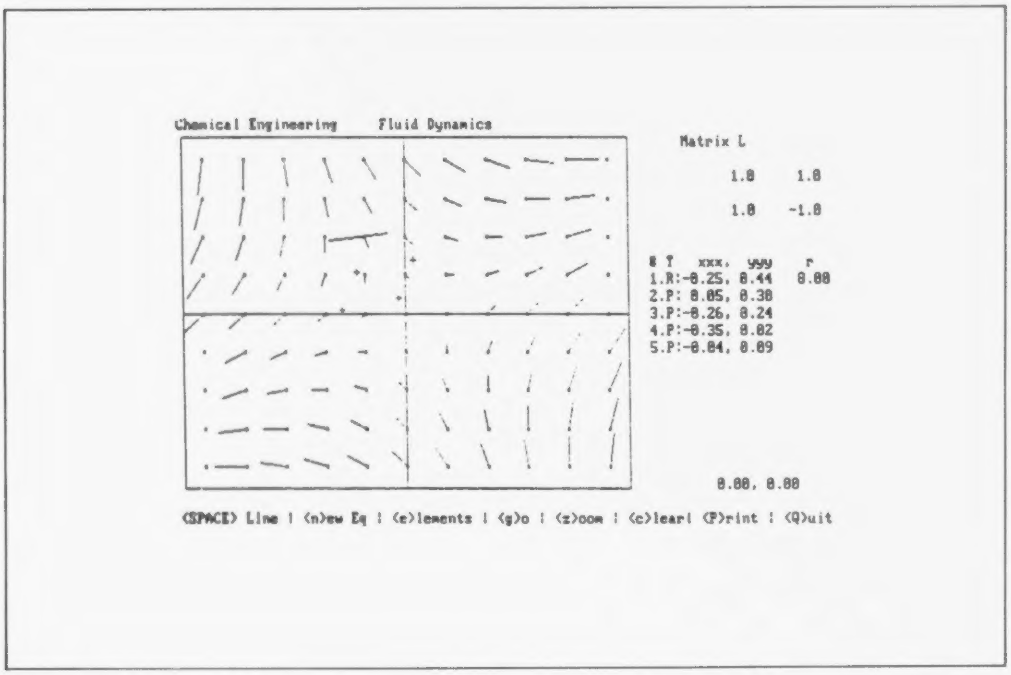
```
Matrix L
1.0   1.0
1.0  -1.0
```

Figure 2.b



```
Chemical Engineering     Fluid Dynamics

                                          Matrix L

                                                1.0     1.0

                                                1.0    -1.0

                                       # T   xxx,  yyy     r
                                       1.R:-0.33, 0.69   0.00
                                       2.P:-0.06, 0.41
                                       3.P:-0.28, 0.41
                                       4.P:-0.28, 0.13
                                       5.P:-0.06, 0.13

                                                  0.00, 0.00

<SPACE> Line | <n>ew Eq | <e>lements | <g>o | <z>oom | <c>lear| <P>rint | <Q>uit
```

Figure 2.c



```
Chemical Engineering     Fluid Dynamics

                                          Matrix L

                                                1.0     1.0

                                                1.0    -1.0

                                       # T   xxx,  yyy     r
                                       1.R:-0.25, 0.44   0.00
                                       2.P: 0.05, 0.30
                                       3.P:-0.26, 0.24
                                       4.P:-0.35, 0.02
                                       5.P:-0.04, 0.09

                                                  0.00, 0.00

<SPACE> Line | <n>ew Eq | <e>lements | <g>o | <z>oom | <c>lear| <P>rint | <Q>uit
```

Figure 2.d

COMPUTER AIDED SYSTEM ANALYSIS AND DESIGN

Presenter: Bradley W. Dickinson *Princeton University*

INTRODUCTION

This paper covers part of the activity supported by the AEP project "Systems and Signal Processing Microcomputer Laboratory". A new course on computer aided methods in system analysis and design was developed and taught in the Fall 1985 semester. It will become a permanent part of the Princeton EE undergraduate curriculum, being aimed at juniors having a background in differential equations, linear algebra, elementary circuits and electronics, and elementary discrete systems fundamentals. The goal of the new course is to exploit the capabilities of PC/AT workstations for supporting software tools that greatly expand the scope of an introductory systems course. By breaking the constraints imposed by problems and examples having easily determined analytical solutions, computer aided software tools bring the issues of real practical importance to the fore: what are the limitations of low order and linear models; what approximations methods are used to study complex systems; what approaches may be used to optimize performance of complicated dynamic systems?

Dynamic systems, modeled by sets of differential equations, are commonly introduced to electrical engineering students having a background in circuits -- transient response and frequency response -- and basic electronics -- diodes and transistors. The traditional academic focus on closed-form analytical methods does not provide a familiarity with the computer-aided analysis and design methods prevalent in electrical engineering practice. The goal of the new course is to introduce systems basics with a focus on issues that are fundamental to computer-aided techniques for analysis and design. As taught for the first time, course topics included: linear algebra and differential equations review; state equations, input-output equations, time response and frequency response; qualitative behavior of systems, including stability, phase-plane analysis, linearization, and describing functions; discretization of systems to obtain difference equation models; parameter optimization, optimal control and dynamic programming. Inclusion of some (stochastic) simulation and system identification material is planned for future offerings of the course.

COMPUTATIONAL SUPPORT

A set of five guided computer projects and one student–selected independent project were assigned during the course. These projects were carried out in a laboratory consisting of 12 networked PC/AT workstations, equipped with EGA and display, 2 Proprinters, and a 7372 (six–pen) plotter. The main software package used was PC–MATLAB, a commercial package available in a special university classroom kit, written and marketed by The MathWorks, Inc., 158 Woodland St., Sherborn, MA 01770. PC–MATLAB is a package of reliable numerical linear algebra routines with numerous supporting enhancements for systems analysis and design and for signal processing, graphics display, etc. It is an interpreted language with a high level, natural syntax that students found very easy to use. PC–MATLAB has extensive on–line documentation facilities that allow students to learn its features quickly. The one drawback of PC–MATLAB that is irksome, namely the long running time of programs involving loops and iterations, was not found to be a problem when used in this course, mainly because PC–MATLAB so drastically reduced the time and effort required to write programs employing reliable numerical routines. (The instructor's patience was tested once or twice during the development of the course projects!)

The five projects assigned to students covered the following topics: Introduction to PC–MATLAB and Linear Algebra Calculations; Bode Plots and Frequency Response of Linear Systems; Sample–and–Hold Discretization and System Time Response; Phase Plane Portraits for Linear Systems; Complicated Behavior of Nonlinear Systems – Limit Cycles and Chaotic Solutions. The projects required the use of a good range of PC–MATLAB's capabilities, emphasizing the writing of programs (command scripts), and utilizing its convenient graphics utilities. The nonlinear systems studied in the last project were piecewise linear, thus enabling a PC–MATLAB analysis.

Some additional software was developed for the purposes of the course. Two items of special note are the driver VDINETPL.SYS and the program cplot.m, together with the associated programs plot.for and read.c. The latter three were developed by R. Alexander and Y. Lim to allow PC–MATLAB to exploit the EGA and display, providing high resolution graphics output through the use of VDI drivers. The IBM Plotting System was used in conjunction with Professional FORTRAN to write the plotting programs; the Lattice C Program and the

PC–MATLAB program provide necessary interfaces between the plotting program and PC–MATLAB.

A problem encountered in using our plotting package only became evident when we were unable to use the 7372 pen plotter across the PC–Network connecting its server to remote network computers. It was discovered, but never found to be documented, that the difficulty lies with the VDI driver supplied for the 7372, VDIPLSIX.SYS. Using the VDI Device Driver Toolkit, a suitably modified device driver was written by J. Blake, enabling plotting from any network computer on the one network plotter.

Additional systems software, mostly in the form of DOS batch files and PC–MATLAB command scripts, was written to ease the housekeeping problems that arise in trying to operate a network of computers while preserving system configurations, network configurations, etc., and to use PC–MATLAB in a network environment.

As mentioned above, at the end of the course students were required to choose a topic for investigation in the microcomputer laboratory. Much of the network housekeeping software and the new VDI plotter driver were written as student projects. Other students investigated computer aided control systems design, computer aided circuit analysis, and graphics packages that are to be used in the Systems and Signal Processing Microcomputer Laboratory in conjunction with other EE Department courses. Also explored were applications of PC–MATLAB to optimization problems and to linear predictive signal analysis. All students presented their projects in a meeting of the class held for that purpose, and all seemed to find the project to be a valuable part of the course experience. In coming years it is planned to devote more time in planning the projects and in meeting with students before the presentations to make the experience even more worthwhile.

CONCLUSIONS

As expected, the use of suitable computational tools in an introductory systems course offers many opportunities to expose students to phenomena that they can learn from by interaction in a way that is not possible in the context of textbook problems and examples or in the context of programming projects using a language such as FORTRAN. Students have been able to carry out projects that would have been too complicated and time consuming to do by hand (even if that were possible) or by writing numerical analysis programs. The focus of the course was kept, quite properly, on gaining an understanding of dynamic system fundamentals.

Choosing a set of good projects to illustrate systems concepts that are of central importance to computer aided analysis and design techniques is a real challenge. It can be expected that some significant changes in the standard pedagogical approaches to this material will result from the growing availability of tools such as those used to implement our project. After only one time teaching the new course it is clear that much remains to be done in refining and distilling the new approach, and just as clear that the rewards for course development efforts will be greater relevance, to say nothing of greater appeal to the students.

ACKNOWLEDGEMENTS

PC–MATLAB was obtained for the Systems and Signal Processing Microcomputer Laboratory through the support of the New Jersey State Department of Higher Education, 1985 Technical/Engineering Education Grant Program, Contract 86–990650–1480. Also supported were acquisition costs for several other state of the art software packages for computer aided design and analysis to be used in a total of four EE Department courses, all of which also use the facilities of the Laboratory provided by the IBM AEP program.

COMPUTERIZED INSTRUCTION IN ENGINEERING THERMODYNAMICS

Presenter: **S. A. Klein** *University of Wisconsin*

Coauthors: **S. A. Klein** *University of Wisconsin*
W. A. Beckman *University of Wisconsin*

Classical thermodynamics is a fundamental course in all engineering disciplines. Thermodynamics is based primarily upon two postulates, the first and second laws which have been formulated in terms such that they are of universal generality. Despite this apparent simplicity, thermodynamics is a conceptually difficult subject. The student is introduced to concepts such as heat, work, energy, entropy and reversibility which have interpretations differing from those used in common language. These concepts must be carefully applied in order to solve engineering problems. Most students will agree that thermodynamics is the most difficult subject they encounter in their undergraduate program.

Thermodynamics is a problem solving discipline. To learn thermodynamics, one must work problems. Most thermodynamics text books include a large selection of problems at the end of each chapter and many authors provide answers for some or all of the problems. The answers provide the student with immediate feedback. If the student's solution differs from that in the text, he likely has made an error. The problem, however, is that the student may not know where in the course of the multiple–step solution the error occurred.

To help students learn how to solve thermodynamics problems, we have developed CP/THERMO (Computerized Problems in Thermodynamics). The program, which runs on IBM microcomputers, is highly interactive and centered around problem solving. It is intended for students in their first and second engineering thermodynamics courses. It could also be useful to the student or professional engineer who has already completed his thermodynamics studies, but who wishes to review the material.

CP/THERMO is not intended to replace either the lecture or the text; it is intended as a supplement to conventional homework. Its primary purpose is to provide a mechanism for students to learn the subject matter on their own more effectively and more enjoyably than if he or she were to attack conventional homework problems with paper and pencil. The program offers several major advantages over the conventional approach. First, in the event that the student is stuck on a particular step of the problem, HELP is available. The HELP processor is the central focus of the program. Repeated requests for help first provide a hint, then a specific suggestion, and finally the answer to the intermediate problem step so the the student can proceed. The alternative with the conventional approach is either to seek advice from the instructor or, more likely, to discontinue working on the problem. Second, the computer eliminates some of the work associated with solving problems, such as looking up properties in tables and algebra. Freed from these tasks, the student can concentrate on the concepts involved in the solution methods.

In designing CP/THERMO, we have attempted to make the program both easy to use and non–restrictive so that the student may solve problems in any manner he chooses. The first–time user ordinarily requires a few minutes of instruction to become familiar with the format and commands. This learning process is aided by a tutorial problem which can be run at the student's option. The tutorial steps through a typical problem, automatically supplying responses and comments concerning the program operation; the student need only press the space bar after viewing each screen to become familiar with operation of the program.

Upon startup, the student is presented with a menu listing topical areas which currently include Concepts and Definitions, First Law Problems, Second Law Problems, Vapor Power Cycles, Vapor Refrigeration Cycles, Reciprocating Engines, Gas Turbines, Psychrometrics, and Chemical Reactions. After selecting one of these topics, a second menu appears offering a choice of problems areas for the selected topic. For example, the problem areas for the topic Concepts and Definitions currently are Temperature and Pressure, Properties of Gases, Use of Property Tables, and Mass Balances.

After a problem area selection is made, the program randomly chooses among the available problems and displays the problem statement. Figure 1 shows the display of a Second Law problem, typical of those encountered in a first semester course, concerned with the analysis of a gas turbine. A figure indicating state points and other information then appears as shown in Figure 2. Both the problem statement and the figure can be directed to a printer. At this point, a menu appears listing the possible operations for this problem as shown in Figure 3. (This simple problem requires only one thermodynamic system. In more complex problems, the menu in Figure 3 would be preceded by a menu allowing the student to choose the thermodynamic system.) The student chooses the operation by entering a number. Alternatively, the student can select any of the options listed on the bottom line of the screen. These options, available at all times by entering the first letter, serve the following purposes:

HELP is for the student who is stuck and does not know how to proceed. Examples of the extensive help capabilities, which form the heart of the program, will be discussed later in this paper.

FIG causes the figure to be re-displayed.

TEXT re-displays the problem statement.

STAT provides a status report listing the quantities that are asked for in the problem statement and summarizing the known information. The optional score and elapsed time are also displayed. The STAT screen at the start of the gas turbine problem appears in Figure 4. Information is automatically filled in as the solution of the problem progresses.

DEFN defines the symbols and units used in the problem as shown in Figure 5. All problems operate in either English or SI units and with either sign convention for work into the system.

PROP is used to obtain information on thermodynamic properties. Depending on the problem, this processor operates in one of two distinct modes. In the automatic mode, the PROP processor will automatically provide the values of all thermodynamic properties for the selected state from the steam tables, R-12 tables, ideal gas tables, etc. provided that sufficient information is already known to fix the state. If sufficient information is not yet available, the known and unknown property information is summarized and the student is invited to enter the value of any unknown property. He may enter H for help at this point to obtain specific information relating to determining the property values. The automatic mode, used in most of the problems typical of a second thermodynamics course, spares the student the task of looking up and interpolating information from property tables. In the manual mode, the student must look up the property data as he would in conventional homework problems, and enter the values. The program checks these values and thereby tests the student's ability to obtain thermodynamic property data.

QUIT provides a graceful means of exiting the program or starting a different problem.

In our classes we encourage students to follow a pattern when they solve thermodynamics problems in which they choose a system and then apply the general statements of mass, energy and entropy balances and then simplifying them for the chosen system. CP/THERMO attempts to reinforce this method of problem solving. In this problem, the student selects an Energy Balance by entering a 1 from the menu in Figure 3. In this case, the screen appears as shown in Figure 6 and the student is prompted to enter the appropriate equation. The equation entered by the student is checked; any algebraic equivalent of the correct equation is acceptable. Editing keys for insertions, deletions, etc. are available so that the equation does not have to be retyped during reentry in the event that first response was incorrect. If the student is unable to correctly enter the program, he can ask for help. The Help processor will first indicate whether there are any missing terms or sign errors in the students response. Additional requests for help will provide the equation without signs and finally, the correct equation.

After the correct equation is determined, the known and unknown information is summarized as shown in Figure 7 and the question "Do you wish to enter values for any of the unknown quantities?" appears. The value of h1, for example, could be entered at this time by responding with a Y and then entering the value of h1. CP/THERMO eliminates the need for the student to use a calculator in determining quantity values since the program will accept numerical values or expressions, including expressions which contain symbols of known quantities. For example, in this problem all of the following responses would be accepted by the program as the correct value for h1.

```
h1 = 367.2
h1 = .24 * (1600 - 70)
h1 = cp * (T1 - 70)
```

The student would find, in working this problem, that the values of h2 and W can not be determined at this point since sufficient information is not yet available. A request for help at this point produces "Try to find s2". A second request for help results in "Try the ENTROPY BALANCE". After selecting the entropy balance, additional requests for help would remind the student that the turbine can be considered to be adiabatic and reversible, and that the specific entropy of states 1 and 2 can be determined using ideal gas relations. The necessary relations will be displayed, followed by the entropy values, if still more help is requested.

The problem solution continues in this manner with the student choosing the operations he wishes to perform in any appropriate order. The program forces the student to solve thermodynamics problems in an organized manner. When all of the quantities asked for in the problem statement have been identified, the program congratulates the student with a short tune, and then prints a final copy of the problem statement, figure, and the completed status report.

The HELP feature is, by far, the most important and complex part of the program. There are often many equally correct paths to a solution of a thermodynamics problem. The HELP processor keeps track of the students progress and it is designed to provide information on request which is specific to the path the student has selected. With the HELP processor, the student can never get stuck, since the needed intermediate answer and/or equation will ultimately be provided if HELP is continually requested.

Overuse of HELP can be kept down, however, by the scoring feature. The student's score is decreased by an amount dependent on the quality of the information obtained from the HELP processor. Hints reduce the score by only one point, but up to three points may be deducted depending on how much information is provided.

Our teaching experience with CP/THERMO over the last two years has been very encouraging. Copies of the program are available at our college microcomputer laboratory. The students may either use the college facility or they may copy the program for use on their own computer. We require students to complete four or five problems during the semester, as a supplement to conventional problems from the textbook. In this manner, the students are exposed to the program and many choose to do more problems than required. Each semester, we have solicited student's comments on the program. The vast majority of the comments have been very favorable. During the first year, the students' comments led to significant improvements in the program. Many of the students indicate that the program is fun to use and that it helps train them to be systematic in their problem solving method. However, we have no way to determine if students are learning thermodynamics more thoroughly as a result of using the program.

There are currently very few educational computer programs for the advanced (junior, senior or graduate level) engineer. Development of such programs requires the combination of expertise in the subject area, programming talent, and, as we have found, a very substantial time investment. It is unclear to us at this point whether the developers of educational software of this type can ever realize a financial gain for their efforts, due to problems with illegal copying and distribution. However, interactive computerized instruction provides a powerful new educational tool, which in spite of its problems, will surely be a direction higher education will take in the future as microcomputers proliferate among students and software becomes available.

Version 5.1 PROBLEM SLB111 01/26/86

A gas turbine has been chosen to produce power at a factory for minor tasks. Air enters the turbine at 1600 F and 70 psia and is expanded to 14.7 psia.

Assuming the turbine dperates ideally, and that the air behaves as an ideal gas with cp= .24 BTU/lbm-R and R=.0685 BTU/lbm-R.

Determine:
a) the exit temperature in F
b) the specific volumes at the inlet and outlet
c) the output work per unit mass flow.

The reference state at which both h and s are zero is 70 F and 14.7 psia.

FIGURE 1. Typical problem statement

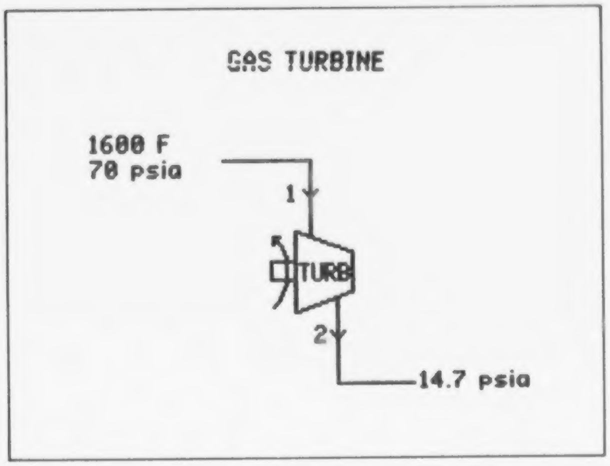
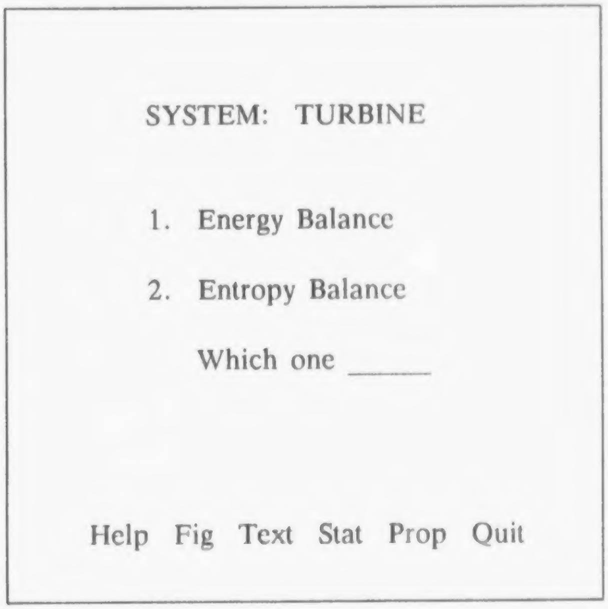


Figure 2. Typical Schematic.



SYSTEM: TURBINE

1. Energy Balance
2. Entropy Balance

Which one ______

Figure 3. Typical Choice of Operations.

```
Time 24 min 26 sec              STATUS                    Score = 100

FIND: T2 v1 v2 W

                          1         2
T F                    1600    ______
P psia                   70      14.7
v ft^3/lbm           ______    ______
h BTU/lbm            ______    ______
s BTU/lbm-R          ______    ______

cp = .24 BTU/lbm-R     R = .0685 BTU/lbm-R     Tsurr = 530 R
```

FIGURE 4. The Status Screen

DEFINITIONS

PROPERTIES
(STATE NUMBER FOLLOWS SYMBOL)

T – temperature {F}
P – pressure {psia}
v – specific volume {ft^3/lbm}
h– specific enthalpy {BTU/lbm}
s – specific entropy {BTU/lbm–R}

OTHER QUANTITIES

cp – constant pressure specific heat {BTU/lbm–R}
R – gas constant {BTU/lbm–R}
Q – heat input per unit mass {BTU/lbm}
W – work per unit mass {BTU/lbm}
SP – entropy production per unit mass {BTU/lbm–R}
Tsurr – temperature of the surroundings {R}

Work into system is a negative value.

FIGURE 5. Definitions Used in this Problem.

ENERGY BALANCE

A steady-state energy balance on the TURBINE (neglecting potential and kinetic energy terms) has the following form:

Symbols which may appear are: W Q h2 h1

Input the equation in the box.

Editing Keys --> <-- Insert Delete Home End Ctrl-Home Ctrl-End Esc

FIGURE 6. Equation Input Mode.

ENERGY BALANCE

A steady-state energy balance on teh TURBINE (neglecting potential and kinetic energy terms has the following form:

$$h1 - h2 + Q - W = 0$$

KNOWN:

UNKNOWN: H1 H2 Q W

We have 4 unknowns and only one equation.

Do you wish to enter values for any of the unknown quantities? (Y/N) __

TIME = 62 MIN 41 SEC SCORE = 94

HELP FIG TEXT STAT DEFN PROP QUIT

FIGURE 7. Equation Summary Screen

DEVELOPMENT OF AN EXPERT SYSTEM FOR COMPUTER-AIDED INSTRUCTION IN PROCESS CONTROL

Presenter: **Manfred Morari** *California Institute of Technology*

copyright © California Institute of Technology, 1986

Coauthors: **Manfred Morari** *California Institute of Technology*

Roland E. Heersink *California Institute of Technology*

Anthony Skjellum *California Institute of Technology*

ABSTRACT

This paper discusses a software approach to integrating expert system concepts with conventional engineering design software. This approach provides a convenient teaching tool and learning/working environment suitable for users of different skill levels and interests. First, we discuss the background, motivations, philosophy and goals behind designing an expert system for control design. This approach avoids multiple versions of software tools for research, education and industrial purposes. Second, we outline the status of our prototype ROBEX system currently under development. In this vein, two examples are provided, the first of which is designed for general audiences.

The ROBEX concept illustrates how both existing and new engineering design software can be integrated into a single environment suitable for a variety of purposes. Repetition of effort is minimized and user-friendliness is enhanced.

CHAPTER 1: GOALS AND MOTIVATIONS

I.1 INTRODUCTION

The process of integrating new theoretical developments into course material and computerized courseware is a difficult task. Often, researchers generate programs which are quite suitable for their particular interests but have no hope for use as teaching tools because they totally ignore the issues of software robustness and user-friendliness. A number of software tools must be used together to generate a problem solution. Therefore, it is necessary for programs to communicate and interact as a package, however loosely.

This chapter discusses our goals, motivations, philosophy and approach to creating an expert environment which effectively integrates diverse algorithms. The general structure of the proposed system is also discussed. We anticipate that this system will be useful for computer-aided design and learning. The real power of the system as a teaching tool is highlighted. We emphasize the software's ability to be set for different backgrounds and needs. We also indicate the effort-saving nature of the system as a means for quickly introducing new theoretical findings and procedures into courseware. Our area of emphasis is control system design but the software concepts can readily be applied to other areas.

I.2 GOALS AND MOTIVATIONS

Our goal is to produce a tool specific to control system design, the ROBEX system, which offers a flexible and extensible environment in which to house existing and newly-developed research results. Flexibility means that users can alter aspects of the system to reflect their needs. Extensibility suggests that concepts can be readily added or deleted without changing the basic structure of the software. By providing a single environment, we reduce the incremental effort required to introduce new concepts, procedures and engineering methodologies.

With such a tool in hand, we will immediately be able to promulgate current research results as practical tools for a variety of audiences. The first audience is students. We will be able to offer a convenient environment where students are encouraged to experiment and hence develop greater intuition. The second audience is practicing engineers. A tremendous gap exists between practical engineering procedures and theoretical techniques. This gap can most effectively be narrowed by offering research results and insights in a black-box configuration. Otherwise, the cost for learning new theories is too great a barrier for the vast majority of engineers. The third audience is researchers. First, we offer ourselves an integrated environment for using our own results. Second, we will be able to share easily these results with others not keenly aware of our work.

It is anticipated that non-experts using the ROBEX system will benefit from the ability to use design methods without an extensive theoretical background. The system will allow the user to specify the control problem in an intuitive fashion. The user will be able to select his or her level of interaction with various software procedures during program execution. The system will attempt to provide a solution which meets the entered specifications. Upon completion, standard control outputs will be available. Furthermore, the system will offer a report detailing its design decisions.

The system should prove useful to students within the control design discipline. To the uninitiated, the system will allow completion of complex design tasks without complete understanding of all the theory involved. As progress is made, the student will be able to interact more closely with the design task by specifying a greater particular approaches to the problem. This approach to learning will allow the student to develop intuition and a deeper understanding of the theory on a gradual basis, all through the use of one software tool.

Similarly, expert users will have the option to set criteria which allow them to concentrate on their areas of expertise while minimizing their effort with respect to mundane design issues. They will be able to specify classes of methods and indicate emphases. By specifying a greater degree of interaction with the software, the expert user will be able to offer "advice" in instances where the software encounters difficulties.

I.3 APPROACH AND PHILOSOPHY

The naive and expensive approach to satisfying the needs of all audiences is to produce a number of versions of each software tool. Such approaches result in multiple (conceptually incompatible) software packages for representing the same body of theory. However, it is usually difficult to justify the development of multiple brands of each software tool, especially for research faculty. We present an approach to short-cut this dilemma. We propose an expert system as a means for computer- aided instruction and a convenient abode for the algorithms representing new theoretical developments. Such a system can be designed to adjust to students, researchers or practicing engineers of radically different backgrounds and needs.

Artificial Intelligence and Expert System terminology is used with abandon in current literature. We adopt a very down- to-earth approach to our expert system project. First, we rely heavily on existing theoretical results from the control system discipline. We avoid the popular probabilistic approach often applied in expert systems but which produce results of dubious reliability. Second, we avoid esoteric programming systems to produce our system. The C programming language is used to represent the data structures, algorithms and input–output routines of the decision–support code while Fortran continues to figure in the engineering algorithms. These choices help assure acceptable performance on an IBM PC–AT or similar computer system while retaining a viable upgrade path to Unix–based systems. We also avoid the cost of recoding existing engineering software in a new language.

Researchers using our system may code new results in either Fortran or C. However, added attention must be paid to input–output and error handling considerations. Front–ends of stand–alone programs are removed and the remaining subroutines are integrated into the tree structure of the expert system as new tools. We demonstrate in chapter II that the extensibility of the expert system can be achieved without significant loss of execution speed.

I.4 SYSTEM STRUCTURE

The ROBEX system consists of four major sections: kernel, input–output, design and analysis routines and information system. These sections are logically connected as shown in Figure 1. The kernel provides top–level orchestration of the other subsystems to effect smooth system operation. The input– output routines will permit access to standard graphical control outputs while enhancing user–friendliness. The design and analysis routines are the C– or Fortran–coded design techniques integrated into the system via the information system.

The unique aspects of the ROBEX system are a consequence of the information system. This information system is composed of two parts: an information network and the data interconversion and handling routines. The information network allows the system to evaluate and discard various design alternatives. The data interconversion routines allow intuitive control system specification. The data handling routines permit extensible integration of numerical design routines into the network.

The ROBEX system contains an information network which allows it to systematically attack a design problem. The information network is a set of tree structures containing design procedures and rules. These trees are recursively traversed in an attempt to evaluate and discard various design alternatives and, if possible, arrive at a feasible solution to the current problem. The association of rules at each branch in the tree allows the incorporation of design heuristics into the system.

The data interconversion routines allow intuitive specification of the control problem. These routines require the development of new theory to relate more rigorously the intuitive specifications to those required within the control design field. Further comments about these theory gaps are made in chapter II.

The data handling routines of the ROBEX system provide a means of passing numerical data (arrays, files, etc.) to design routines located in different areas of the network. This system also accepts data files from various other systems for conversion and input into the system. The routines will be able to minimize transfer to and from the storage media while performing file handling tasks for the design and analysis routines as portably as possible.

I.4 CONCLUDING REMARKS

We have outlined the general concepts and requirements for the construction of an expert system tool specific to the control system discipline. Target audiences have been identified and student use for such a system outlined. The ROBEX approach to engineering design software integration and system structure was also discussed. In the next chapter, we discuss these issues with particular reference to the version of the ROBEX system under development.

CHAPTER II: CURRENT SYSTEMS DEVELOPMENT

II.1 INTRODUCTION

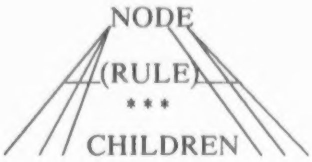
The application of expert system techniques to control system design offers new opportunities and challenges. In the previous sections, we discussed the theoretical goals and ideas concerning the integration of process control synthesis and analysis methodology into an expert system structure. In this chapter, we discuss the practical considerations involved in the initial prototype version of the ROBEX system.

This chapter overviews the structure developed to date for the ROBEX system and presents two illustrative examples. The focus of the chapter is on software issues rather than on theory development. Readers seeking detail on control applications and the theory research mentioned previously are referred to the second example.

II.2 CURRENT SYSTEM DESCRIPTION

As previously advertised, the ROBEX system consists of four major sections: input–output, design and analysis routines, information network, information system, and controlling kernel. To date, the major emphasis has been on the development of the information network. Therefore, the remainder of this section describes that aspect of ROBEX.

The heart of the ROBEX system is the information network which links the various design and analysis routines and heuristic rules. This network is constructed of a set of nodes all of the general form shown below:



The complete network consists of the specified design methods linked together in a "tree" or "forest" construct. The "root" (of the tree or forest) acts as the starting point for all solution procedures.

Within the tree, each node can have any number of children and can link to them with a rule of any "and/or/not" combination. Alternatively, appropriate C functions can be used to implement more sophisticated criteria. To allow use of existing control design routines written in Fortran, nodes can link also with any Fortran subroutine by means of a C header function. Fortran routines typically become "leaves" in the tree or forest. The C header function converts results provided by the Fortran code into a truth condition for that node.

Each node in the network is constructed on the basis of four attributes which follow:

identifier
descriptor
number of children, children identifiers
rule

The information for all nodes is housed conveniently in a text file. This file representation makes changes to the network a simple matter of editing a file and using a canned build routine.

At runtime, the network is evaluated recursively, in a right-to-left manner, starting at the bottom left hand corner of the tree and working up. The "deduction motor" checks the rule associated with the particular node under consideration before evaluating each child. When the value of the node can be determined using the rule, the remaining children are abandoned and the motor moves on. This minimizes execution time, since no more evaluations are done than absolutely necessary.

Several other features incorporated into the system alsominimize runtime. Undo runtime overhead is avoided by keeping the tree structures fixed during execution. As mentioned, convenient off-line procedures for making structural changes are available. To avoid repetitive execution of certain sub-trees or operations fundamental to several design methods, an appropriate time-saving mechanism is offered. This is done by allowing appropriate nodes to point to the same children. The savings afforded by this capability can be significant in control system design.

II.3 EXAMPLES

Two examples are outlined to describe concretely the network structure. The first of these examples illustrates system features via an animal identification package and is suitable for all audiences. The second example describes the prototype control design system under construction and requires a sound control design background of the reader. These examples were not chosen for control-relevance but rather to offer insight into the software mechanisms.

II.3.1. ANIMAL IDENTIFICATION PACKAGE

The animal identification package allows the user to identify an animal based on a collection of pictures. The network is structured so that the user is asked a series of questions about the animal under scrutiny, based on previous responses. This situation is analogous to having a node access (query) a Fortran control design routine for results leading to the determination of a controller meeting the specifications.

Several node definitions are given below as examples. The form given below is identical to their normal text-file representation.

# Node 22
brbear
The animal is a brown bear.
4,large,furry,horned,longneck
rule22, (1 and 2 and (not 3) and (not 4))

# Node 118
large
See if the animal is large.
2,small,medium
rule118, ((not 1) and (not 2))

# Node 101
small
Is the animal small in size relative to a human?
0
descrule

In the rules defined above (eg., rule22, rule 118), the numbers (1,2,...) refer to the children of the associated node, in the order in which they appear. In Node 101, the rule "descrule" is a simple C routine which outputs the node descriptor to the user as a question. This routine also interprets the user's response and sets the node's value accordingly.

From the given examples, we can see that a brown bear is a large animal, an animal which is neither small nor medium in size relative to a human. A brown bear also has the characteristics of being a furry animal that has neither any horns nor a long neck. The text-file format is designed for easy modification and not for user visualization of the resulting tree.

To identify an animal, the user is asked a series of questions as the "deduction motor" recursively traverses the network, while attempting to evaluate the root. If the root evaluates to true, we have identified one of the animals in our database. This corresponds to finding a feasible control design. Otherwise, no animal is identified, which is analogous to an infeasible control design problem.

II.3.2. PROTOTYPE CONTROL DESIGN SYSTEM

The tree structure associated with the ROBEX prototype single-input, single-output proportional-integral-derivative (SISO PID) control design system currently under development is depicted in Figure 2.

The input phase of the prototype system permits the user to specify the problem in a limited fashion. The plant to be controlled is entered as a transfer function with the associated uncertainty represented as either a set of parameter ranges for the transfer function coefficients or a multiplicative error expressed as a function of frequency. Allowable closed loop performance specifications include settling time and an optional disturbance transfer function Pd(s).

After starting at the root, the first node evaluated by the "deduction motor" is the left-most bottom node, "Calculate eps, K, W2" which is reached by recursive descent. This node verifies that the problem description is feasible. The performance specifications are subsequently converted from settling time and Pd(s) to roll-off rate, eps, maximum peak of the sensitivity function, K, and performance weighting function W2.

Before the "motor" continues on to the next child, the current rule is examined in light of the value of the child just evaluated. This establishes whether further evaluations are required to determine the value of the parent node "Design Method #1." For example, should the "Calculate eps, K, W2" node be found to be false (infeasible performance specifications), no more children need be evaluated, and, in this particular example, the design system will fail to find an appropriate controller.

The next child, "Calculate Regions," computes the smallest uncertainty regions for the specified plant transfer function and associated uncertainty description. If the plant transfer function is defined with uncertain coefficients, the smallest convex polygon is calculated over a range of frequencies to characterize that uncertainty [1]. If the transfer function is given with multiplicative uncertainty as a function of frequency, then a region-based uncertainty description (a set of circles with radius as a function of frequency) is inherent in the description. In either case, a region-based uncertainty description is saved for possible use later in robust stability and performance tests.

The "Design Controller" node is evaluated next and makes use of Internal Model Control (IMC) techniques to design a PID controller [2]. The IMC structure, shown in Figure 3a, inte– grates a plant model, p~(s), and an IMC controller, q(s), into the classical block diagram structure of Figure 3b.

The IMC design procedure consists of two steps. First, the nominal plant description is factored into two parts, p~+ and p~–. The p~+ transfer function contains all non–minimum phase elements of p~ and is constructed to be all–pass by addition of minimum phase dynamics. The second step defines the IMC controller, q, as f/p~–, where f is a low–pass filter selected to yield a robustly–stable closed–loop system. The IMC controller is related to the classical feedback controller, c, by the relationship:

$$c = q \,/\, (1 - p{\sim}q)$$

In order to have the IMC design methods result in a PID controller, the full order model is simplified to second order by model reduction techniques. Weight functions emphasizing important model characteristics are obtained by control–relevant model reduction methods. These weight functions are subsequently used in a standard weighted model reduction algorithm to determine an appropriate low order model [3]. Finally, the resulting controller is tuned using rules available for plant models common in process control [4].

Given a controller, appropriate stability and performance tests are performed. Both nominal and robust stability are required to satisfy the "Test Stability" node, as per rule1. This rule allows the appropriate robust stability test to be made. Depending on the plant description, the robust stability test is based on either multiplicative uncertainty or region calculations. Similarly, the "Test Performance" node requires only the appropriate test to be made; either based on mu criteria or on the calculated uncertainty regions [5],[1].

Future work on control system design implementations will allow more intuitive problem specification by the user. These systems will incorporate research which must provide a more direct relationship between the time domain specifications meaningful in industrial practice and the frequency domain specifications required for most rigorous design techniques. Existing theory does not offer a convenient mechanism for relating response envelopes in the time domain to frequency–domain uncertainty descriptions. Therefore, current work involves not only the adaptation and creation of software technology, but also the development of new theory.

II.4 COMMENTS AND CONCLUSIONS

Future work can be classified into two distinct areas. First, much additional work is needed in the development of the ROBEX system. Specification and development of input–output software is especially important as, without user–friendliness, non–experts will be hopelessly lost. Also, the information system must be able to manage the wide variety of files and data types inherent in the system. A further aspect of the information system will be the ability to exactly or approximately interconvert different types of data and specifications. The latter capability will depend largely on the new theory to be developed.

The second area involves the development of this new theory. It is necessary to relate the specifications and uncertainty descriptions given in the time domain to those required within the frequency domain. Time–domain specifications are what a non–expert could reasonably provide while frequency domain specifications are needed by rigorous control design techniques.

III.1 CONCLUSION

In this paper, we have discussed a software system named ROBEX. In chapter I, we discussed the motivations, and goals behind designing an expert system for control design. We touched on the need for a convenient environment for housing research results in a way useful to experts and non–experts alike. Furthermore, we indicated that this choice of theory presentation removes the need for multiple incarnations of software bearing the same theoretical basis. Therefore, the ROBEX system offers a mechanism for quickly providing new research results as tools for learning and practical design.

In chapter II, we outlined the current status of the ROBEX system. We provided details specific to the information network including two examples. The first example was not control–related to permit general understanding. The second was a SISO PID design example. This latter example is representative of a small portion of the complete control design tree.

While the preceding discussion has focused on development of a control system design tool, the general structure of the ROBEX system should prove useful as a framework in which to link intelligently methods specific to any field. The ability to integrate existing C and Fortran routines into the information network makes it especially appealing to engineering design applications. Therefore, while we are creating a specific expert package, the software technology could be carried over to uses in other disciplines.

REFERENCE LIST

1. D.L.Laughlin, K.G.Jordan & M.Morari, "Internal Model Control & Process Uncertainty: Mapping Uncertainty Regions for SISO Controller Design," Accepted for publication, **Int. J. Control**, 1986.

2. C.E.Garcia & M.Morari, "Internal Model Control: A Unifying Review & Some New Results," **I & EC Process Des. & Dev.**, V21, No.2, 1982.

3. D.E.Rivera, M.Morari, "Internal Model Control Perspectives on Model Reduction," 1985 ACC.

4. D.E.Rivera, M.Morari & S.Skogestad, "Internal Model Control 4: PID Controller Design," **I & EC Process Des. & Dev.**, V25, No.1, 1986.

5. J.C.Doyle, "Analysis of Feedback Systems with Structured Uncertainty," **IEE Proceedings**, Part D, V129, No.6, 1982.

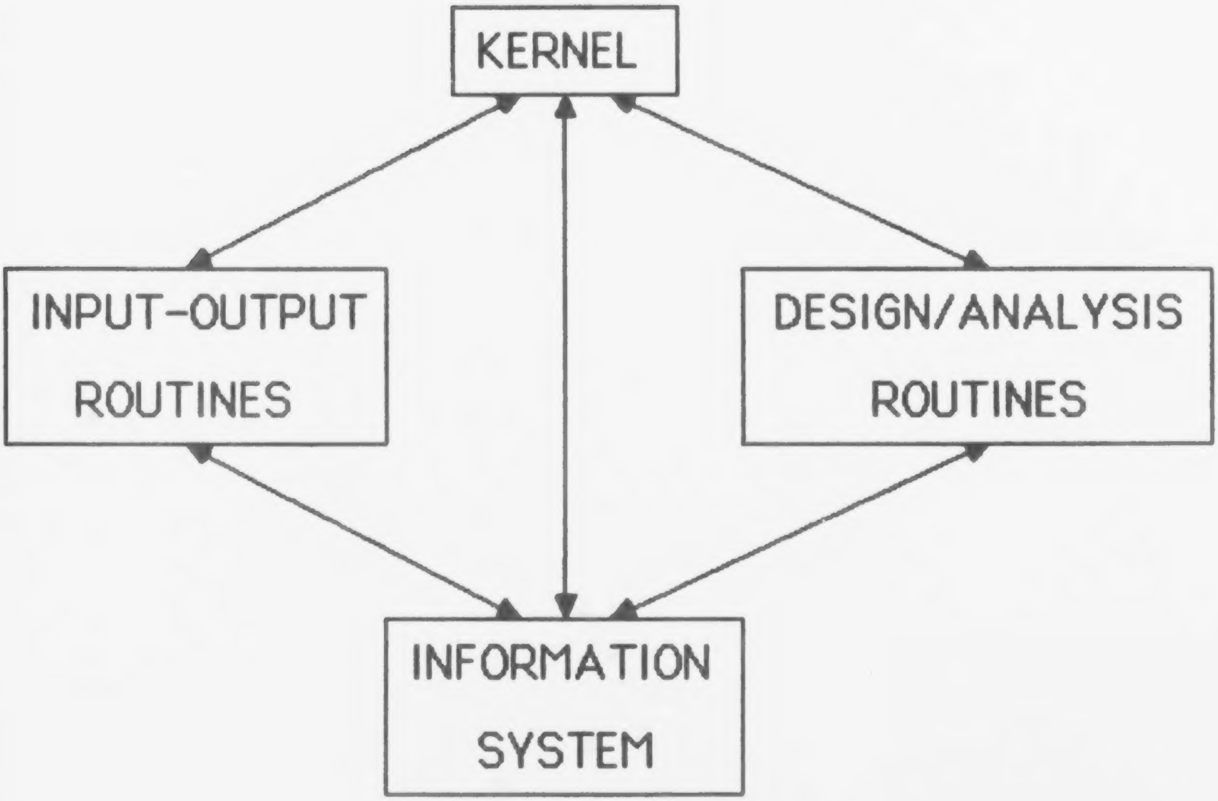
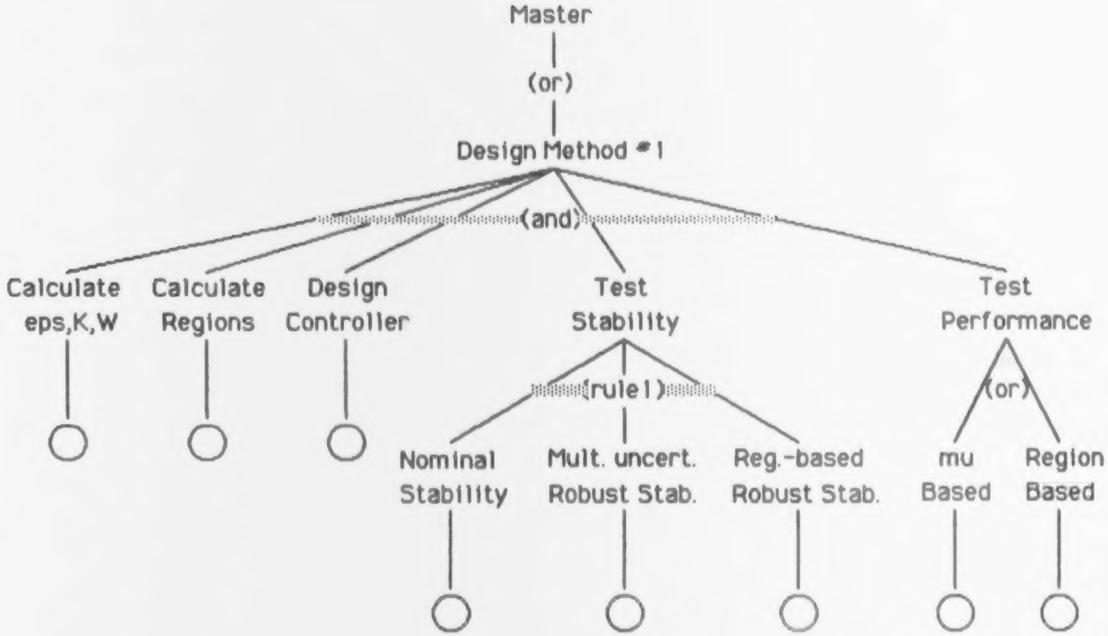


Figure 1. System Structure



○ = Appropriate FORTRAN Interface Routine (C function)

* rule 1 = (1 and (2 or 3)).

Figure 2. Network Representation of Prototype PID Control Design System.

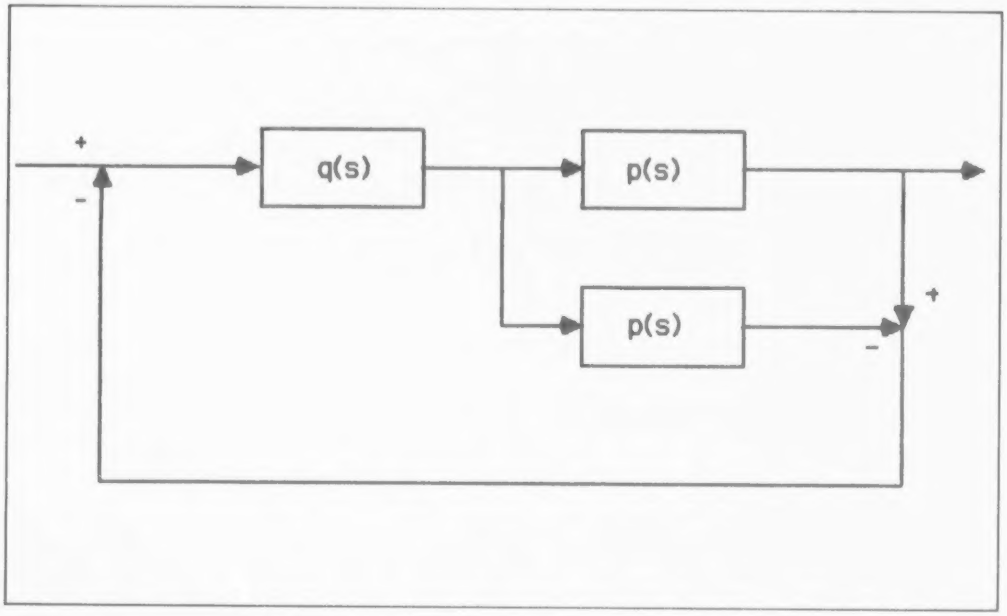


Figure 3.a Internal Model Control Structure.

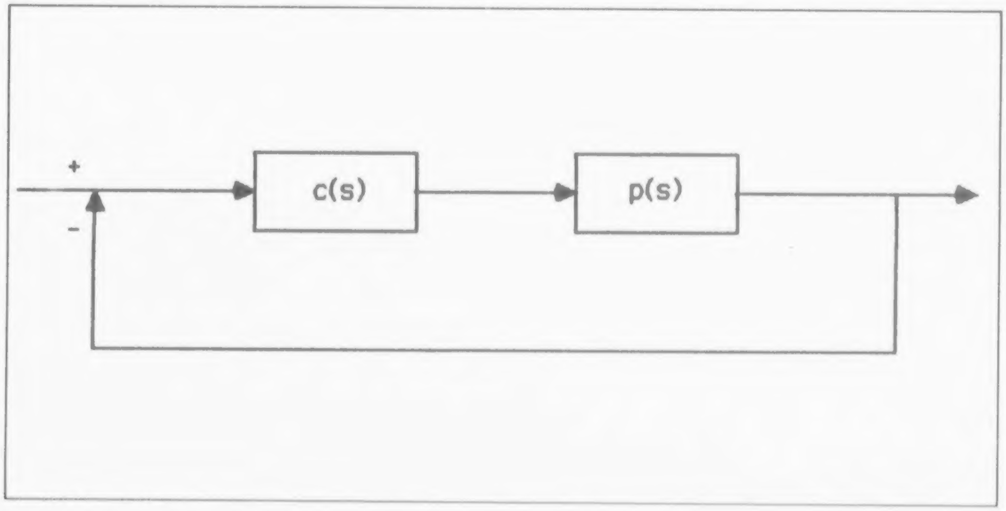


Figure 3.b Classical Feedback Structure.

APPENDIX 1: CONFERENCE OVERHEADS

DEVELOPMENT OF AN EXPERT SYSTEM FOR COMPUTER-AIDED INSTRUCTION IN PROCESS CONTROL

by

Roland E. Heersink
Anthony Skjellum
Manfred Morari

California Institute of Technology

April 1986

GOAL:

To develop an extensible and flexible process control design system for use by students, researchers and practicing engineers.

OUTLINE OF TALK:

1. Overview of applicable expert systems technology.
2. Overview of existing process control design systems.
3. Description of our current work, the ROBEX system concept.

EXPERT SYSTEMS - OBJECTIVE

- to embody expert-level knowledge in a computer, replacing the expert

- many applications already do this in very specific domains

- to do this in a flexible manner for a realistic case is non-trivial

EXPERT SYSTEMS - TYPES

- Typically rule-based
  - if (conditions) then (results)
  - also called production systems

- Two basic types of systems
  - synthesis expert systems
  - analysis expert systems

- Methods of attack
  - forward chaining

    results ⟶ conditions
  - backward chaining

    conditions ⟶ results

EXPERT SYSTEMS - OPERATION

- At each step in the evaluation process:
    - all applicable rules are collected
    - all but one eliminated
    - result executed

- Repeat the above until:
    - problem is solved

  or:

    - no more "if" parts of the rules are satisifed

      → no problem solution

EXPERT SYSTEMS - FEATURES

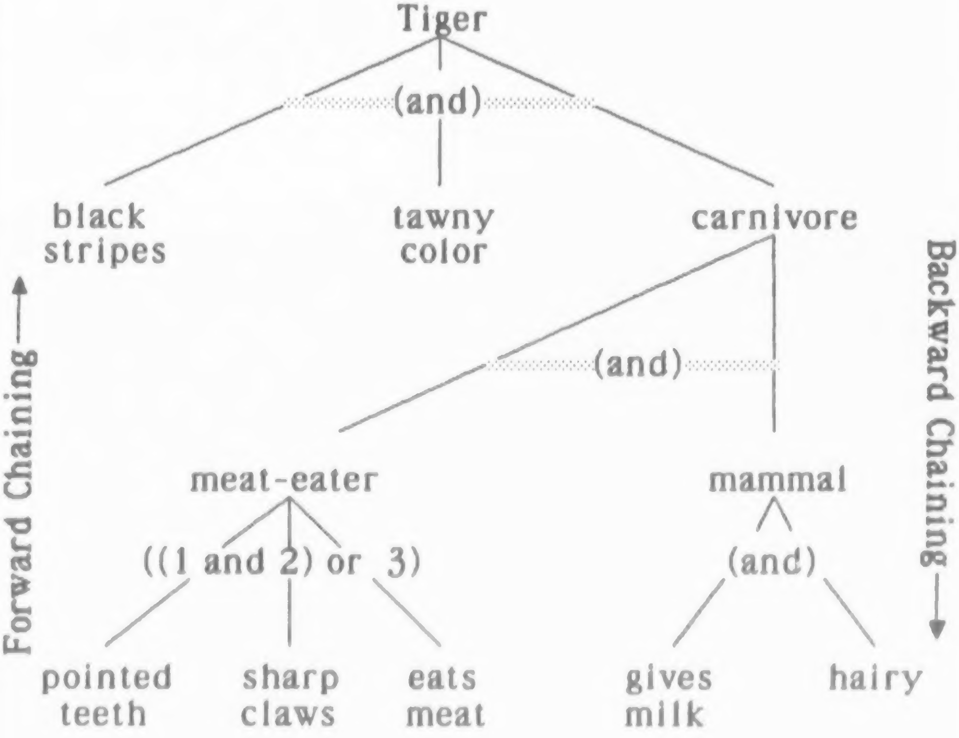
- Able to answer questions on how & why decisions were made
    - uses historical records of rules triggered to do so

- Natural language capabilities
    - enhances user-friendliness

- If desired, uncertainty measures can be incorporated into rules

EXPERT SYSTEMS - DATA REPRESENTATION

- One way to represent data within the computer with network structures:



EXPERT SYSTEMS - PROBLEMS

- Results only as good as data put in
  - everything is false until proven true
  - limited databases can cause problems
- Computer follows rules like a novice i.e. goes "by the book"
- Run times can be long
- Most expert systems do not deal well with extensive numerical calculations

PROCESS CONTROL DESIGN SYSTEMS

- Most packages available today:
    - require a sound control background to operate
    - are generally not user-friendly
    - seem user-hostile to most new users

PROCESS CONTROL DESIGN SYSTEMS

- Many different methodologies exist by which to complete a control system design

- Many different ways of specifying a system are possible

- Choices between the above options are not always obvious, especially to the neophyte

PROCESS CONTROL DESIGN SYSTEMS

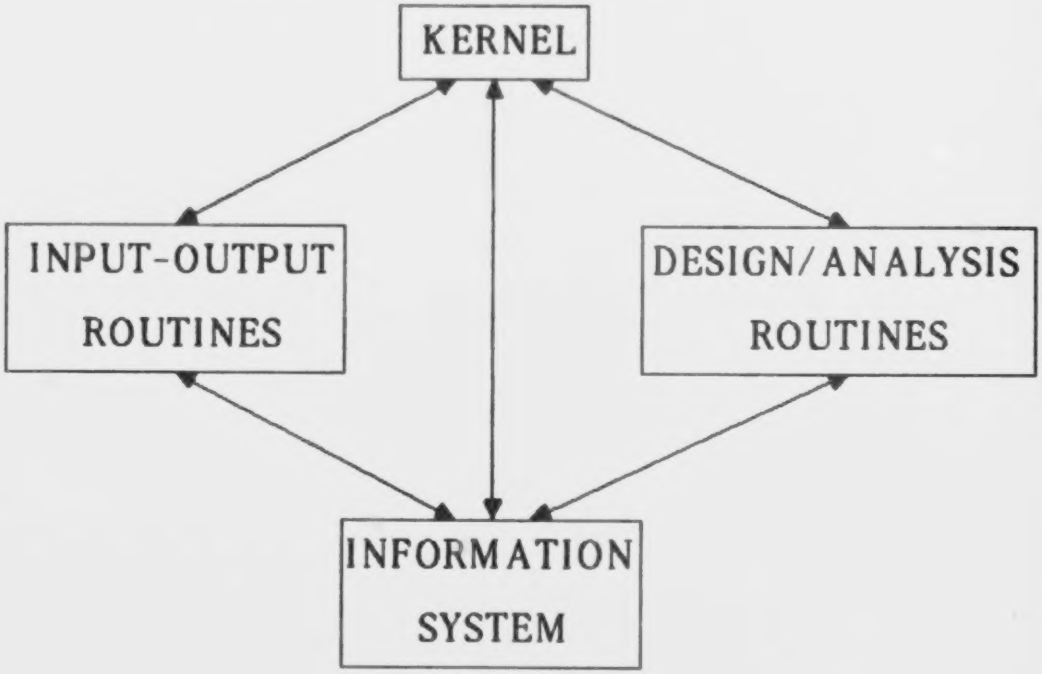
- Ideally, a control design system should:
    - work effectively with users of different skill levels
    - be extensible and flexible
    - allow intuitive types of input
    - "grow" with users as they develop understanding

THE ROBEX SYSTEM

- The ROBEX system is designed to:
    - work effectively with users of different skill levels
        - allow new users to complete designs immediatly
        - allow experts to concentrate on their area of expertise
    - System makes use of expert system concepts, where useful
    - System is designed for IBM PC-AT under XENIX in C

      ⟶ portability

THE ROBEX SYSTEM - STRUCTURE

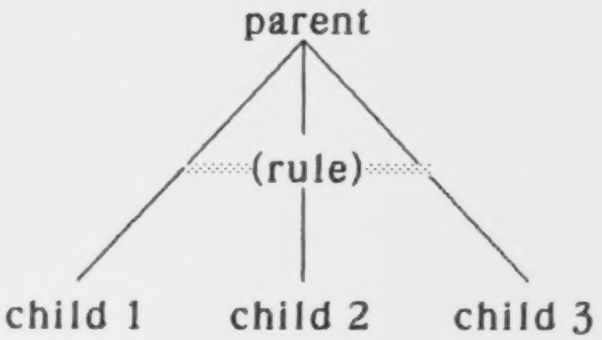


THE ROBEX SYSTEM - INFORMATION SYSTEM

- Information system consists of three parts:
  - data network
    - contains all links to design methods and logic
  - data interconversion
    - converts various data to allow more intuitive problem specification
  - data handling
    - permits numerical data to be passed within the tree structure

THE ROBEX SYSTEM - DATA NETWORK

-Data network is constructed of nodes of the form:



parent

(rule)

child 1 child 2 child 3

- Some possible rules might be:
  - and
  - or
  - (1 and (2 or (not 3))). etc.
  - any C function

THE ROBEX SYSTEM -DATA NETWORK

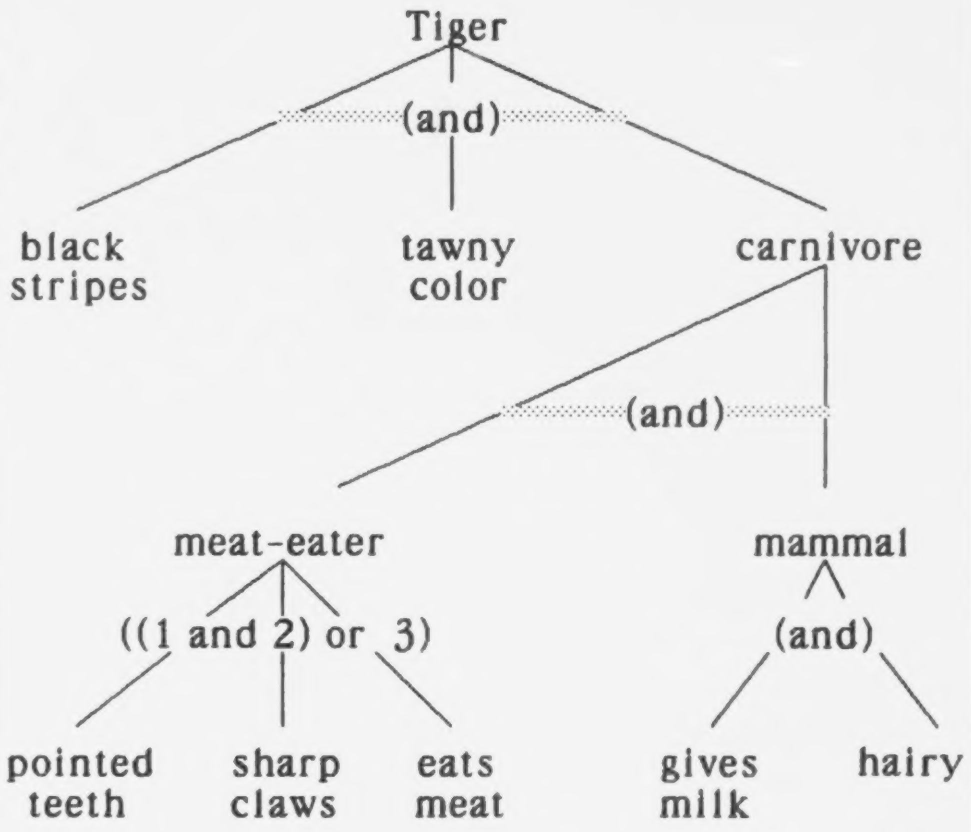
- Network is built from a text file with entries for each node as below:

node identifier
node descriptor
# children, identifiers
node rule

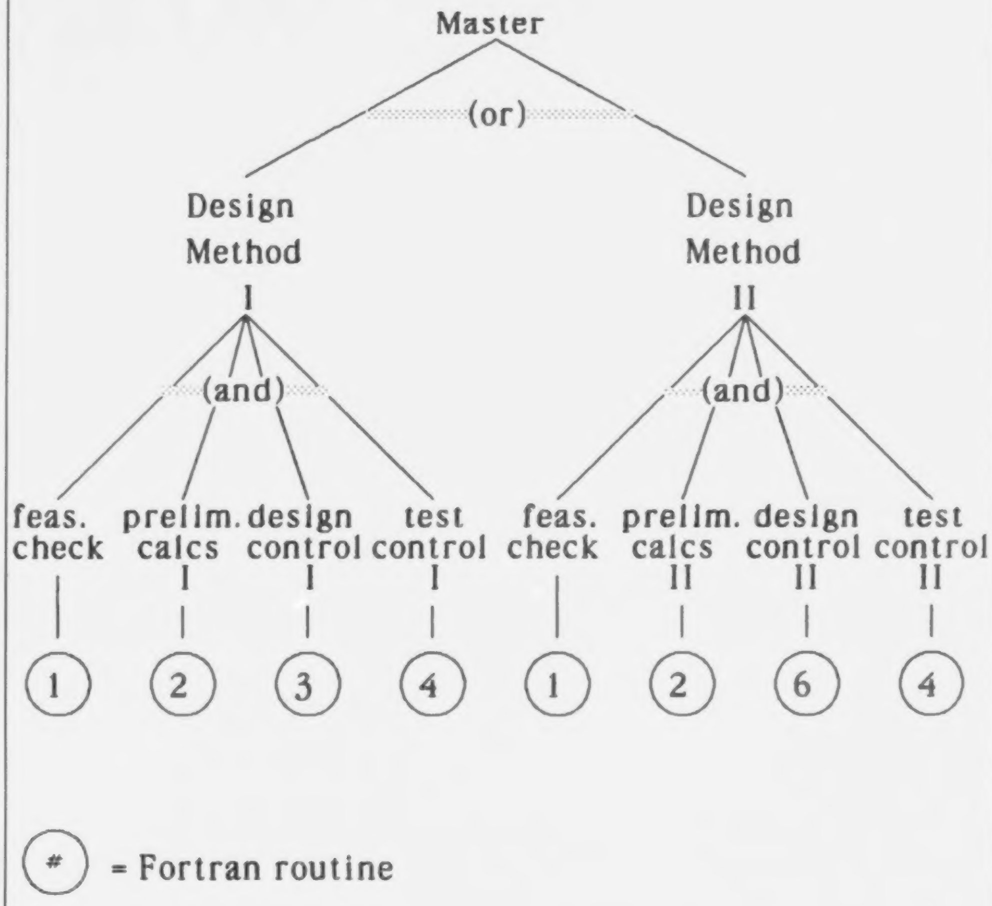
- For the preceding example, the file entries for the parent node would be:

parent
This node is the parent node.
3, child 1, child 2, child 3
rule1,(1 and (2 or (not 3)))

THE ROBEX SYSTEM - DATA NETWORK



THE ROBEX SYSTEM - DATA NETWORK EXAMPLE



THE ROBEX SYSTEM - SYSTEM FEATURES

The ROBEX system is flexible, users can:

- set environment corresponding to his or her skill level
- rule out specific design methods or classes of methods
- set level of interaction with software to be consulted at a given point or when problems arise

- The ROBEX system is extensible:
    - new methods can readily be incorporated into the network

THE ROBEX SYSTEM FOR STUDENTS

- Students will find the ROBEX design tool very convenient as they will be able to:
  - complete design tasks immediately
  - "grow" by increasing interaction with the system as they develop expertise
  - build insight and intuition incrementally
  - use the system as a development environment for new research & homework

INTERACTIVE MICROCOMPUTER GRAPHICS SOFTWARE FOR MODELING GEOGRAPHIC AND TIME VARYING PHENOMENA

Presenter: Daniel P. Loucks *Cornell University*

ABSTRACT

This paper describes some of the software being developed at Cornell University to aid those involved in modeling environmental phenomena having both spatial and temporal dimensions. Such models typically have substantial data management needs. The software, being developed for use on an IBM PC-XT/AT with 8-bit color graphics capabilities, is designed to aid in the entering, editing, displaying, and understnding of both model input and output data. Some applications of the software are presented. Its use for students in classes on interactive environmental modeling with microcomputer graphics will begin at Cornell in the Fall of 1986.

INTRODUCTION

Over the past two decades increasing attention has been given to the development and use of models for predicting and aiding in the management of environmental impacts. Such impacts typically result from regional development activities that affect any or all environmental media: air, land, and water. These impacts vary over space and time, and hence many environmental models require both spatial and temporal dimensions. Managing all the data required and produced by these models has become an increasing challenge, especially as these models become more detailed and more complex.

In addition to the growing need for improved data management procedures, there is the longstanding need to make models that are designed for planning and management more useful to those who could benefit from their use. Improvements would include developing more effective means of using these models to explore alternative development plans displaying their resulting impacts in a relatively fast and easy manner. This suggests work toward developing improved model-user interaction and data input, editing, and display should increase the potential for using models to help synthesize or create as well as analyze alternative future development and management plans or policies.

These arguments together with the improvements in microcomputer technology and graphics digitizing and display capabilities prompted some of us at Cornell to begin the development of interactive computer graphics software in the late 1970's. IBM's Project Ezra at Cornell has given us the opportunity to adapt much of what we have developed, and to continue this development, in IBM microcomputer equipment that is widely available to many professionals, both in and outside universities, involved in environmental modeling. Our experiences as well as some of our results to date will be reviewed and illustrated in the sections that follow.

SOFTWARE DEVELOPMENT GOALS AND EQUIPMENT

The central goal of our work in Project Ezra ia to provide environmental modelers with programs they can use to make their models interactive and to take advantage of color computer graphics for more effective data input, management, and display. Our goal is to assist users in integrating various models and data and to enhance model-user interaction without requiring the user to learn as much about computer graphics as otherwise would be required.

We wish to make it as easy as possible for modelers to develop interactive models that use microcomputer graphics. To accomplish this a set of FORTRAN 77 callable subroutines and functions are being developed and documented for use in building interactive menu-driven graphics programs. All routines and functions are layered on top of a GKS graphics standard, hence anyone having a GKS graphics FORTRAN library will be able to use any of our developed software. Thus to the greatest extent possible we are making our software device independent. Our software is designed to reduce program development time and costs, minimize programming error and, at the same time, provide flexibility in programming a fully interactive, user-friendly model. We have been able to evaluate and improve some of our software through its use by a number of graduate students in their research, but the Fall 1986 semester will see its first use in a classroom setting.

Before taking on the task of developing some of our higher-level interactive graphics software, we evaluated the IBM pro-graphic board for meeting our display and interactive modeling needs. This was inadequate for the videodigitized mapping and transparent coloring we find useful. A VDI interface to our software was developed (to the extent possible), but problems in the VDI drivers, its slow speed, and the fact that it does not conform (on the IBM) to the ANSI standard led us to reject this option also. Our goal is to write device–independent software suitable for any 8–bit RGB 256–color displayable high–resolution color graphics system. Because of our special needs for transparent coloring, having a palette of 256^3 colors from which to select the 256–color look–up table is also desirable. Since IBM does not provide this capability we have purchased a VECTRIX 9–bit VXPCB graphics card and compatible RGB color monitor. Other 8–bit boards could also be used with some minor modification in a few of our programs. Our software is written for applications with or without a digitizing tablet and pen, although the latter is a definite advantage for most interactive modeling applications.

SOFTWARE LIBRARY DESCRIPTION

The software being developed is called Computer–Aided Planning Programmer's Library (CAPLIB). It consists of FORTRAN 77 callable routines supporting data entry from terminal keyboard or pick devices (tablet or joystick), menu definition and selection, color palette editing, complex polygon definition and filling, opaque or transparent coloring, variable size screen image, storage and retrieval with run–length encoding and decoding, 2–dimensional graphing, surface interpolation and display, and on–line multi–level HELP capabilities. Using CAPLIB, FORTRAN programmers can build color graphic display devices, write device drivers, or learn the use of a complicated set of "low–level" graphic functions.

Among the features now available in CAPLIB are:

1.) Menu Page Construction and Maintenance Routines:
menu stacking and tracking, drawing and definition, highlighting and selection

2.) Color Manipulation Routine:
color interpolation, color look–up table editing, storage

3.) Color Device and Graphic Attribute Routines:
inquiring and setting color, character size, spacing, font, line style, transparency

4.) Terminal Handling Routines for Graphic and Non–Graphic Terminals:
bell, clearing, inputting data, providing prompts, writing text

5.) Drawing Routines in Opaque or Transparent Modes:
filling lines, areas, 2–D graph labeling

6.) Digitizing Routines:
picking positions, colors, image registration, potentiometers

7.) Image Encoding/Decoding and Display/Storage Routines:
read data, display data, write data

8.) On–Line Help Routines:
information retrieval, start–up and exit routines

9.) Work Session Recording and Playback Routine:
keeping session journal files

10.) Miscellaneous Routines for:
a. initializing graphics devices and defining desired configuration,
b. obtaining current data and time,
c. waiting a specified amount of time,
d. scaling maps and computing areas and distances,
e. surface interpolations, contouring,
f. image highlighting,
g. showing color palettes for user selection,
h. determining minimum and maximum values of an array,
i. converting character strings to real or integer numbers and vice–versa,
j. other miscellaneous routines.

In addition to the above, other programs and files have been prepared to make it easy for the programmer. Include files of all CAPLIB program variables and parameters, and programs for beginning and ending interactive programs are available. A number of example programs designed to help beginners using CAPLIB have also been prepared.

Each of the routines being developed contains the minimum number of arguments required to carry out the prupose of the routine. This reduces the necessity of changing source codes as capabilities are added. For example, to fill a line using CAPLIB, a call is made:

STATUS = FILINE (X_1, Y_1, X_2, Y_2)

This produces a graphic depending on a number of system control parameters and graphics attributes, including

a) current display device,
b) current color,
c) current line style,
d) current display write channel mask.

Suppose a new graphic attribute, line width, is added and an appropriate default value is given to it. Note that

a) the FLLINE function does not need to change,
b) the existing programs can still be rebuilt and correctly run without any alteration of their source code,
c) the programmer gets a new set of routines which permit the setting of the current line width and the asking of what that line width is.

The above description of our software is merely a brief outline of what is contained in much more detail in the CAPLIB documentation manual [RPA, 1986a].

APPLICATIONS

Perhaps the best means of illustrating some of the software routines is by their combined use in the utility program called MAP EDIT (MAPEDT). Using videodigitized maps of the areas of interest, and colors as specific attributes associated with specific locations on those maps at specific times, one can generate model input and output data and display it quite effectively on these map images. MAPEDT is a program for displaying, editing, and storing these digital map or other images. It can be used for the preparation of color-indexed digital maps, which in turn can be used for spatially-oriented analyses. The MAPEDT program is described in detail elsewhere [RPA, 1986b].

MAPEDT is simply a composit of CAPLIB subroutines and functions, and hence is an example of their combined use. To illustrate this, consider Figures 1 and 2 that show some of the menu pages (without graphic picture displays) from MAPEDT. These figures indicate the programming, using a few of the CAPLIB routines, involved when changing menu pages or items. A detailed programmer's reference manual for CAPLIB is available to aid programmers in completing such operations as illustrated and discussed in this simple example [RPA, 1986a].

MOVING BETWEEN MENU PAGES

Normally a menu page is constructed at the beginning of a subroutine or function. Referring to Figure 1, the MAPEDT menu page is constructed near the beginning of the main routine of program MAPEDT. Movement to a second menu page, MEASURE, now will be explained.

On the MAPEDT page the MEASURE menu item has been selected. It was selected by use of the PKMENU function and was highlighted (outlined) by PKMENU by the use of the MNUBOX function. When PKMENU returns to the MAPEDT main routine it returns a status equal to SUCCEs (defined in an include file called PARAMS>INC) and has, in this case, returned with menu item 3 selected (MNUITM=3). The MAPEDT routine examines MNUITM in an IF,THEN, ELSE block and when it finds that MNUITM equals 3 the CPAREA routine is called which builds and presents the MEASURE menu page.

Among the first operations of the CPAREA funciton is the construction of the MEASURE menu page. The calls made are given in Figure 1. First, the CHCALL function is used with arguements (1,'MEASURE'). This call places the name MEASURE on the stack of menu page names over the top of MAPEDT and RPA$STARTUP. On the menu page header only the top two elements of that stack are reported. The call to CHMNAM with arguments (0,'') blanks out all of the current menu options. Calling CHMNAM with arguments (0,'ANYTHING') will place "ANYTHING" in all the menu positions (except for the last four menu positions which are reserved by the CAPLIB software). Next, using CHMNAM, all the menu options of the MEASURE menu page are loaded and the menu page is ready for display. Menu page display is accomplished by the call to CHMENU. The argument value of 3 tells CHMENU to update both the menu page header and the menu options.

The PKMENU routine does not complete a successful menu selection when users point to a blank menu item. Also, it is necessary to have the current menu options displayed. If, for example, you would have performed a CHMNAM(0,'') call right after displaying the MEASURE menu page, users would have been unable to pick any but the last four menu items. This is true whether or not one has caused the "blanked" menu items to be displayed.

The last operation that function CPAREA performs before returning to MAPEDT is to call CHCALL(−1,"). CHCALL with a first argument of −1 removes the latest menu page name from the stack. In this example, the call removes "MEASURE" from the stack and leaves "MAPEDT" and "RPA$STARTUP." Once the program returns from CPAREA to the MAPEDT main routine, that routine performs the CHMNAM(0,") call to blank all definable menus and then rebuilds and redisplays its menu page as shown in Figure 1. It is not necessary to call CHCALL to place the menu page name MAPEDT on the stack. If CHCALL(1,'MAPEDT') was incorrectly called once more, then the menu page header would report that MAPEDT has called itself.

CHANGING MENU OPTIONS

Sometimes you will not want to change an entire page but may want to alter one menu item to report a status or parameter value. This operation is reported in Figure 2.

The MAPEDT menu page always reports the current color (or attrubute number) with which painting and annotation operations will occur. If the menu item reporting that value is selected (using PKMENU), then MAPEDT allows the user to point to a new color (again using color as an index to attributes). The PKCOLR routine is used for that pointing and once a valid color is selected PKCOLR returns with the color number as COLRID. In this example, a FORTRAN INTERNAL WRITE is used to write the number (as well as some text) into a character variable. That character variable (TXT12) is then used as an argument in the call CHMNAM (8,TXT12). This loads the new menu name into the menu options array. The call "STATUS=CHMITM9*0" causes the displayed menu page to be altered to reflect the new menu text. Updating occurs only for menu option number 8.

The remaining figures illustrate some of the MAPEDT, and hence CAPLIB, capabilities.

GRAPHICS DISPLAYS

Figures 3 and 4 illustrate only some of the numerous types of displays that can be created using CAPLIB. In this case videodigitized maps are being used with transparent coloring to indicate site–specific values of certain attributes (eg. nitrate concentrations in the root zone) at a certain time.

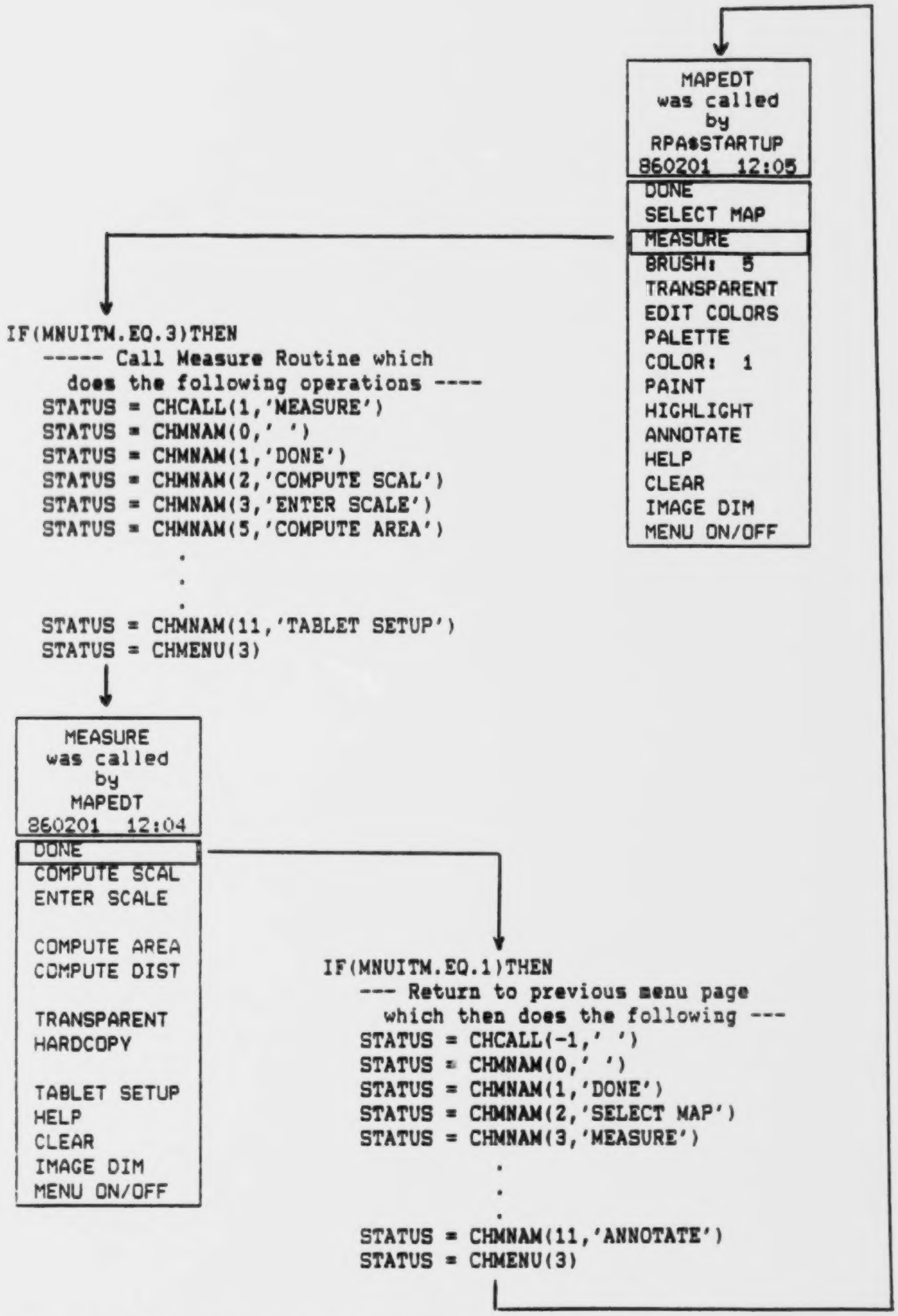
SUMMARY

This paper has been a brief overview of the software being developed to aid those involved in environmental modeling. The software is designed to facilitate the implementation of interactive pre– and postprocessing and the use of computer graphics for data input, editing, and display. With the assistance of an IBM Project Ezra Grant, this software adapted for implementation on a PC XT/AT will be used in a classroom environment beginning in the Fall of 1986. It has already been used by graduate students in their research.

REFERENCES

Resources Planning Associates, 1986a. CAPLIB Programming Guide, Version AT–1.0. Cornell Research Park, Ithaca, NY, March.

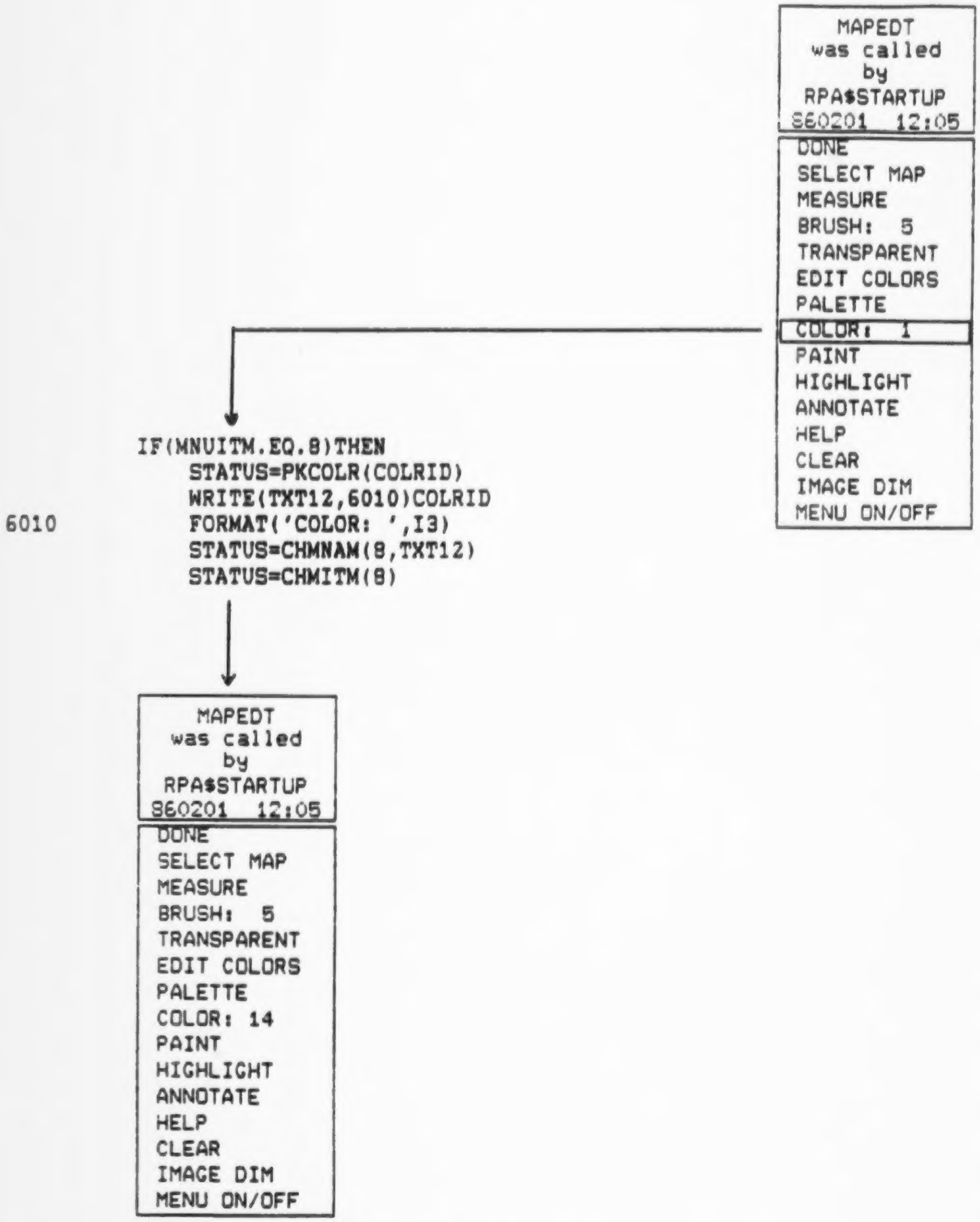
Resources Planning Associates, 1986b. MAPEDT – A Map Editing Program, Version AT–1.0. Cornell Research Park, Ithaca, NY, March.



```
IF(MNUITM.EQ.3)THEN
   ----- Call Measure Routine which
     does the following operations ----
   STATUS = CHCALL(1,'MEASURE')
   STATUS = CHMNAM(0,' ')
   STATUS = CHMNAM(1,'DONE')
   STATUS = CHMNAM(2,'COMPUTE SCAL')
   STATUS = CHMNAM(3,'ENTER SCALE')
   STATUS = CHMNAM(5,'COMPUTE AREA')
                     .
                     .
                     .
   STATUS = CHMNAM(11,'TABLET SETUP')
   STATUS = CHMENU(3)
```

```
IF(MNUITM.EQ.1)THEN
   --- Return to previous menu page
     which then does the following ---
   STATUS = CHCALL(-1,' ')
   STATUS = CHMNAM(0,' ')
   STATUS = CHMNAM(1,'DONE')
   STATUS = CHMNAM(2,'SELECT MAP')
   STATUS = CHMNAM(3,'MEASURE')
                     .
                     .
                     .
   STATUS = CHMNAM(11,'ANNOTATE')
   STATUS = CHMENU(3)
```

Figure 1. Using CAPLIB to move between menu pages: an example.



```
        IF(MNUITM.EQ.8)THEN
            STATUS=PKCOLR(COLRID)
            WRITE(TXT12,6010)COLRID
6010        FORMAT('COLOR: ',I3)
            STATUS=CHMNAM(8,TXT12)
            STATUS=CHMITM(8)
```

Figure 2. Using CAPLIB to change single menu items: an example.

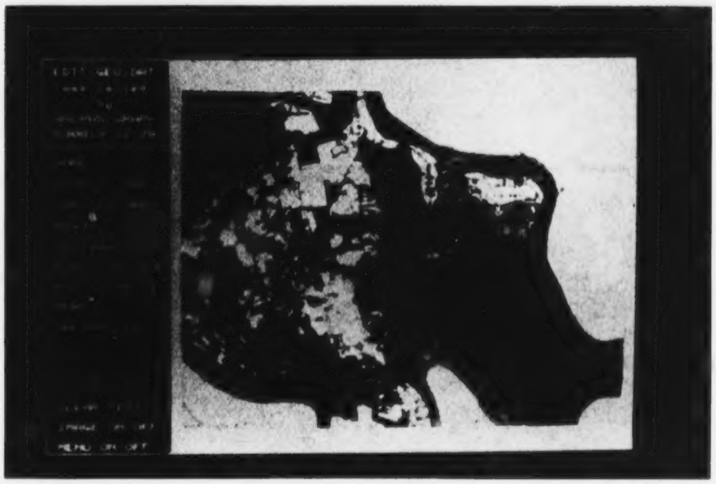


Figure 3.
Videodigitized map with color transparent overlays used to show site-specific attribute values.

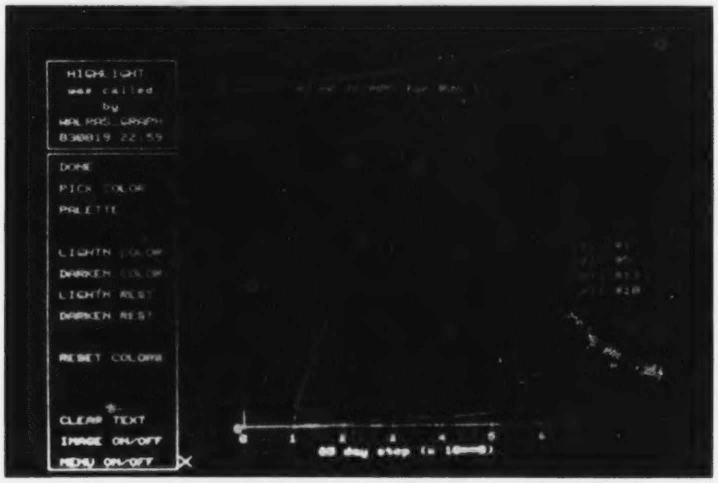


Figure 4.
Same map as in Figure 3 but dimmed to highlight an overlayed graph of a time series of same attribute value that changes over time at a particular site selected by the user.

M C A D A: MECHANICAL COMPUTER AIDED DYNAMIC ANALYSIS

Presenter: Richard C. Coddington *University of Illinois at Champaign–Urbana*

INTRODUCTION

MCADA is a new dynamic simulation program available on IBM/AT and NEC microcomputers for the analysis of constrained, planar mechanisms. Purposes of this paper are to tell you what it does, to describe its primary features and to illustrate its uses in engineering education.

PROGRAM DESCRIPTION

MCADA stands for Mechanical Computer Aided Dynamic Analysis. In a nutshell, MCADA automates construction of the equations of motion, applies constraints consistent with user prescribed revolute and translational joints, solves for the equilibrium forces, integrates the system of equations and displays the response of the mechanical system on the colorgraphics monitor.

MCADA features a menu driven, interactive program, with color graphic displays, on–line help messages to aid the inexperienced user and free format input of the input data set. MCADA's main menu has three distinct options. They are session control commands, viewing commands and data commands.

SESSION CONTROL COMMANDS

MCADA provides five options for session control whereby the user conducts his or her simulation. These options include a static analysis, a kinetostatic analysis, a dynamic analysis, a restart of the simulation or a termination of the simulation. By selecting the static analysis option, the user can assemble his or her mechanism and determine the forces associated with theequilibrium configuration.

For a moment let's digress from a discussion of these options and recall Newton's second law that force equals mass time acceleration. Then exercise of the dynamic option results in the usual dynamic analysis where the user specifies the forces that act on the mechanism and MCADA solves for the resulting motion. On the other hand, one might choose to specify the motion (for example the acceleration) and find the forces that produce that motion. MCADA call this a kinetostatic analysis. The restart option allows the user to restart the simulation from the beginning whereas the stop option terminates it.

VIEWING COMMANDS

Visual presentation of static, kinetostatic and dynamic response of the mechanism is essential to getting an "engineering" feel of the motion of the mechanism. MCADA provides for interactive, color plots of user selected variables and for a display of the mechanism. The DISPLAY option is particularly useful for studying the full range of motion of mechanisms. By displaying the family of positions, the user can examine potential interference problems as well as check for obstructed views.

Among the many choices for display of response, the user may elect to plot the position or speed of user selected links, to plot the forces or torques at user selected spring locations, or to plot the family of positions for a portion of or for the entire simulation.

DATA COMMANDS

As evaluation of the mechanism proceeds, the user may wish to modify some of the parameters basic to the response of the model. MCADA provides for "on–the–fly" changes of design parameters such as spring rate and damping constant. To determine the current values of these parameters, the user simply selects the LIST PARAMETER option and specifies the parameter value of interest. The user then can interactively modify the value of the parameter with the ALTER command and in so doing conduct "What if?" numerical experiments. You see, Hewlett Packard's not the only one that can do this! The print output option lets the user display output on the monitor for hard copies if desired.

Sensing levels are particularly useful in providing the user added control during the simulation. But what are they? An example will help to clarify their meaning. One way to end the simulation is to specify the maximum time for the simulation. However, the user may also wish to sense the values of user selected variables and terminate the simulation when reaching one of the prescribed values of these variables. The ADD SENSE LEVEL permits this type of control. In addition, after reviewing the results thru the display or plot options, the user may wish to continue on with the simulation. The user can then ALTER the sensing levels of the variables and continue the simulation.

APPLICATIONS

MCADA has many practical uses in engineering education. We've tested MCADA extensively over a broad range of applications. I mention a few of these applications to spark your interest. Please help me by imagining where these apply to your activities or where they suggest other related applications of interest to your students.

Some of the applications we've looked at include the analysis of agricultural linkages, the study of ride characteristics of tractors, the handling behavior of articulated vehicles, the design of mechanisms such as compressors, balers, three point hitches and the analysis of low frequency vibrational response of mechanical systems.

In this paper I'll briefly review three examples that demonstrate the power and versatility of MCADA. These examples include the dynamics of a swinging chain, the simulation of a baler–plunger mechanism and the steering behavior of an articulated tractor.

SWINGING CHAIN EXAMPLE

The first example shown in Figure 1 involves the large amplitude motion of a chain with five moveable links connected together with smooth pins at five revolute joints and restrained at the right end by a vertical spring. The diagonal line represents ground with gravity acting vertically downward. You can think of the ground as being a diagonal wall running from the first pin joint on the left to the top end of the linear spring on the right. MCADA always refers to ground as link 1. Consequently, the first chain link, called link 2, is the first moveable body or link.

By using the ALTER command we can "cut" the spring by setting the spring rate and damping constant to zero. We are then ready to do a dynamic analysis. However, prior to this dynamic analysis I added a sensing level to the angle of the first chain link. I set its value to minus 90 degrees. In addition, I picked a simulation time of 1.0 seconds. You can see the family of positions of the chain in Figure 2. The first link rotated thru 90 degrees. In this case, the first link rotated 90 degrees before reaching one second of simulation time.

Notice further, that the last three links rotate together for some time and finally the end link starts to "whip" around similar to a bull whip.

We can put quantitative values to our "feel" of the rotation of the links from the DISPLAY option by using the PLOT option to view changes in the angles of the first, middle and end links. Such plots show the first link starts rotating immediately whereas the middle and end links delay a while before commencing to rotate. In addition the middle and end links rotate a small amount in the counterclockwise direction before starting to rotate clockwise.

Figure 3 shows that the pin forces at the wall on the left end of the chain gradually change at first then rapidly change when the "whipping" action of the end link starts. Thus, MCADA provides useful information on the dynamic forces created at the joints connecting the links forming the mechanism. Armed with this information the designer can conduct "What if?" numerical experiments to avoid excessive dynamic loads that don't appear from a static analysis.

BALER–PLUNGER MECHANISM

The second example involves an agricultural application of MCADA to the design of a baler. Figure 4 shows the model. The first link represents ground. Link 2 is the crank, more commonly called the the flywheel in a baler application. Link 3 is the connecting rod between the flywheel and the plunger used to compact the hay. Revolute joints join the flywheel to ground and to the connecting rod. A translational joint joins the plunger, the right end of the connecting rod and ground. The line on the right represents a nonlinear spring for simulation of the force exerted by the hay on the plunger.

The user specifies motion of the flywheel with a user supplied generator subroutine. A user supplied force subroutine simulates the nonlinear spring rate of the hay. Since this mechanical system has one degree of freedom, one motion generator completely determines the motion of the baler–plunger mechanism. Thus a kinetostatic analysis reveals the forces producing the prescribed motion. Figure 5 shows the movement of the mechanism for one complete revolution of the flywheel.

By using the PLOT option, the user can study response of the mechanism. As the plunger moves to the right it compacts the hay. Figure 6 shows that the force builds up to a peak value then rapidly drops off after reaching maximum compression.

The user is free to select the variables deemed best to reflect the response of the mechanism. Also, the user can interactively select the dependent and independent variables when using the PLOT option.

A student prepared the model for study of hay compaction by a baler as a term project in an Agricultural Engineering course called the Dynamics of Farm Machine Elements. He was able to demonstrate excellent agreement between measured forces obtained from the engineering literature and predicted dynamic forces in the mechanism.

Another student in the same course prepared a model for simulation of three-point tractor hitches with control. His objectives included finding the hitch capacity and the forces in the components, determining influence of hitch geometry on hitch performance, and analyzing the dynamic response of the tractor's hydraulic system.

These as well as the other students in the class felt the learning experience in using the microcomputers for dynamic simulation was very beneficial both in use of the IBM/AT and in using MCADA to study the time dependent, dynamic response of practical mechanisms.

The two students mentioned above combined their projects for an exhibit in our annual engineering open house held the first week in March. They received an award for their exhibit helping our department win the best department trophy.

In the past, analysis of mechanisms focused on studying the motion of linkages at a given instant of time. With MCADA we can conveniently examine what happens to linkages with the passage of time. The complexity of these calculations requires computer aided assistance. MCADA serves well in this capacity.

Later this spring I will use MCADA as a teaching tool in my Agricultural Engineering Design class for the study of commonly used linkages in the design of agricultural machines. It will be very helpful in studying the kinematics of these linkages as well as investigating their dynamic response.

ARTICULATED STEERING MODEL

The last example involves steering response of an articulated tractor. Imagine you're a bird flying over the tractor. Figure 7 shows the outline of the front and rear bogies. The first link represents ground. Link 2 is the front bogie and link 3 the rear one. MCADA permits the user to enter approximate values for the geometry of the mechanism and assembles it during the static analysis. A revolute joint joins the front and rear bogies.

The user steers the articulated tractor by specifying the steering signal with a generator subroutine. In addition the user prescribed two constraints on the lateral motion of the bogies and one constraint on the forward velocity of the tractor. Since this mechanical system has four degrees of freedom, the four motion generators completely determine the motion of the tractor. Thus a kinetostatic analysis reveals the forces producing the prescribed motion.

To get a "feel" of movement of the tractor during the simulation, the user can select the DISPLAY option, enter the number of frames for viewing and press the enter key. By watching the color monitor the user follows movement of the tractor from right to left. Figure 8 shows the "movement" of the tractor for a sinusoidal steering signal. The display of frames resembles to a small degree the animation capability available with MCADA on the NEC microcomputer. Double buffering permits full animation so vital to "giving life" to the simulation of actual response of dynamic systems.

SUMMARY

MCADA is a valuable tool for analysis of two-dimensional dynamic response of constrained mechanisms. Its primary features include automatic construction of the equations of motion, application of constraints consistent with user prescribed revolute and translational joints, solution for the equilibrium forces and position, integration of the system of equations and user selected, interactive displays of response.

Its uses are many. They include design of linkages, design of suspension systems, study of ride response, analysis of steering behavior and evaluation of engine components. MCADA serves as a valuable teaching tool for better understanding of the kinematic and dynamic response of complex mechanisms.

Truly, the panorama of MCADA applications is enormous with the primary limit being the user's imagination.

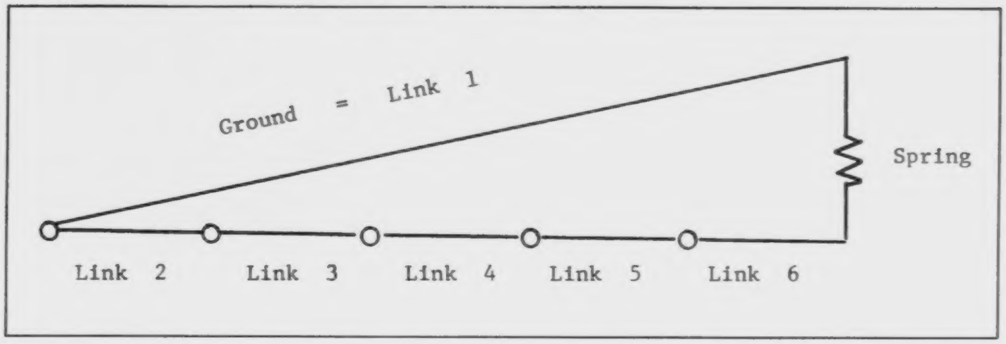


Figure 1. Model of Chain.

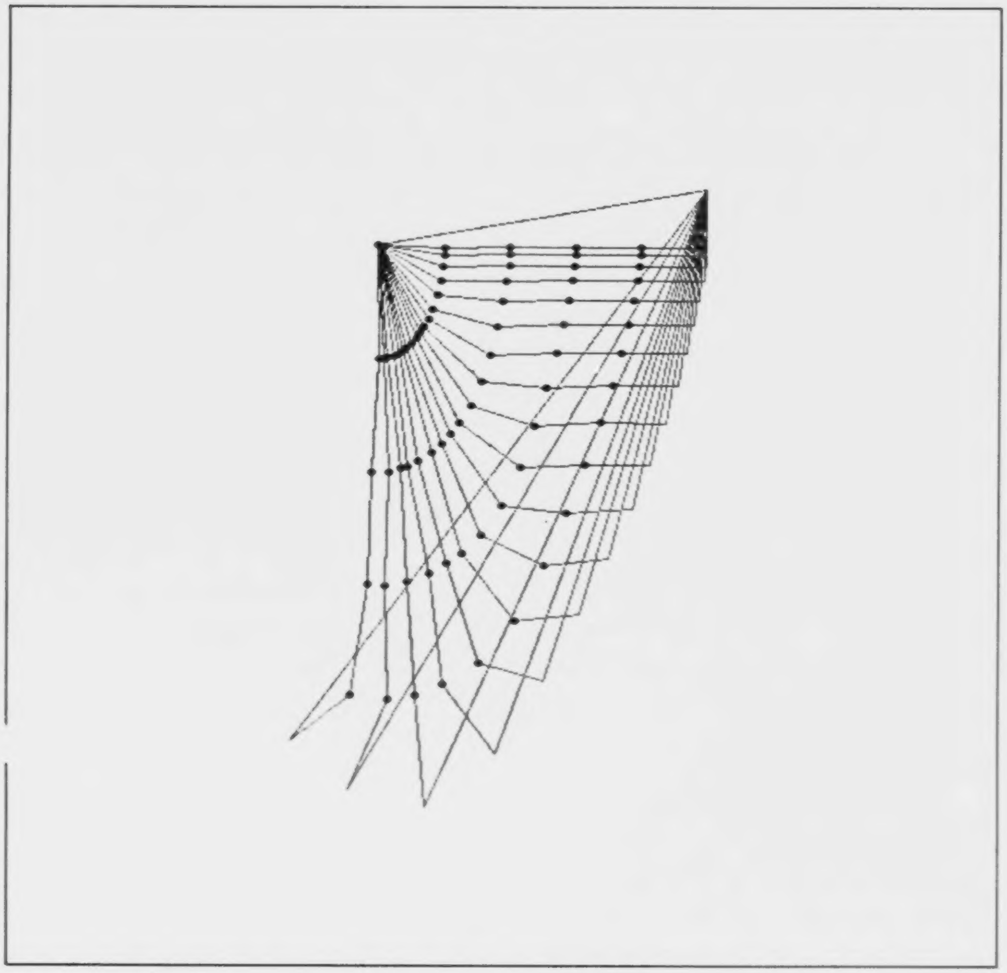


Figure 2. Family of Positions of Chain.

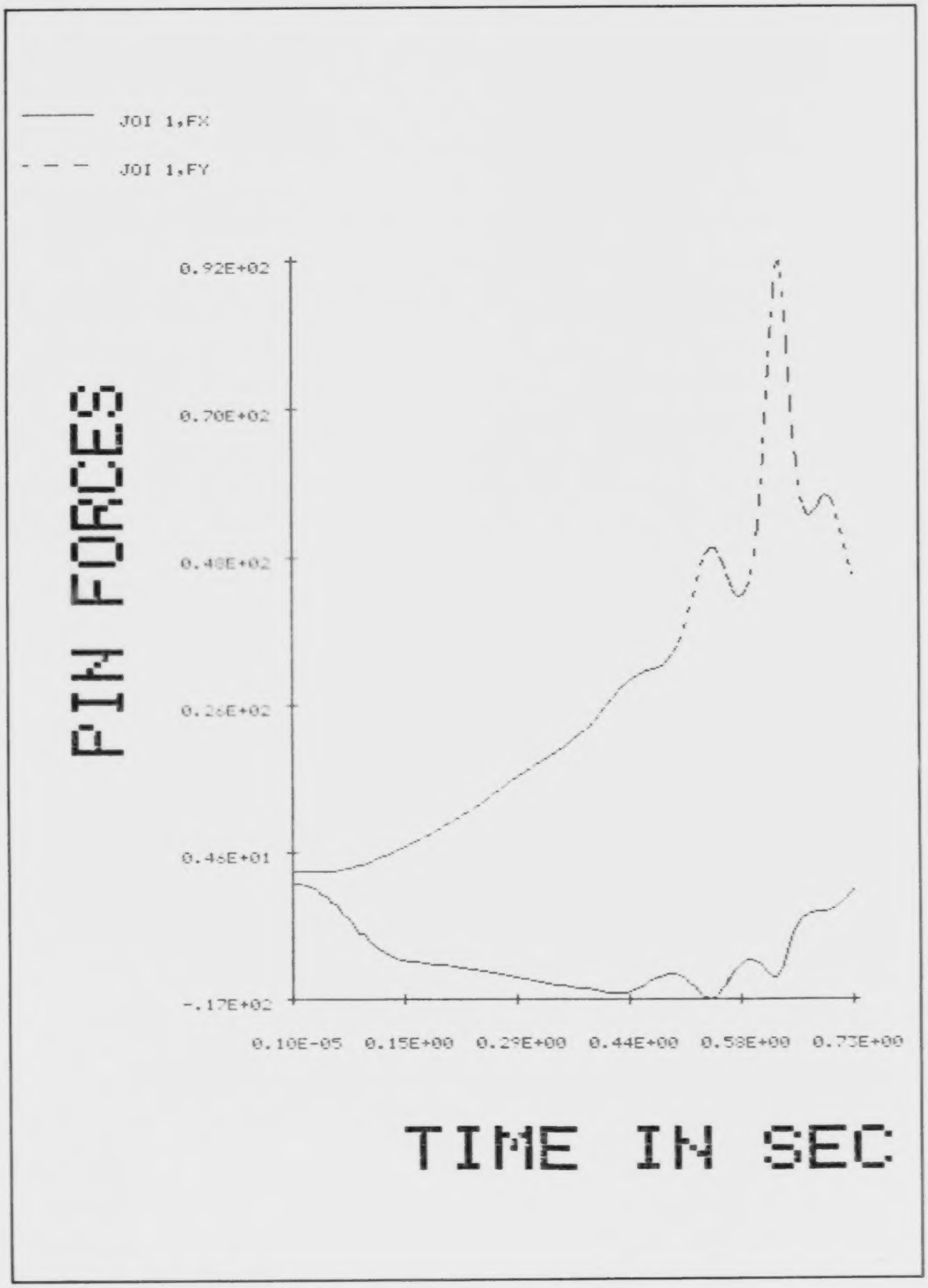


Figure 3. Pin Forces at Wall.

BALER-PLUNGER

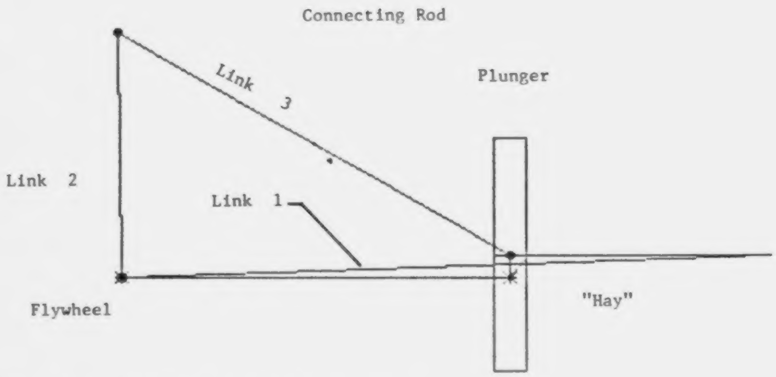


Figure 4. Baler-Plunger Mechanism.

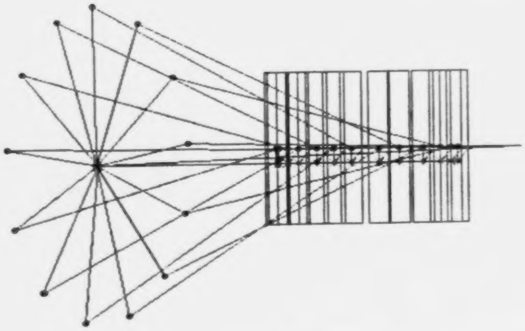


Figure 5. Family of Positions of Mechanism.

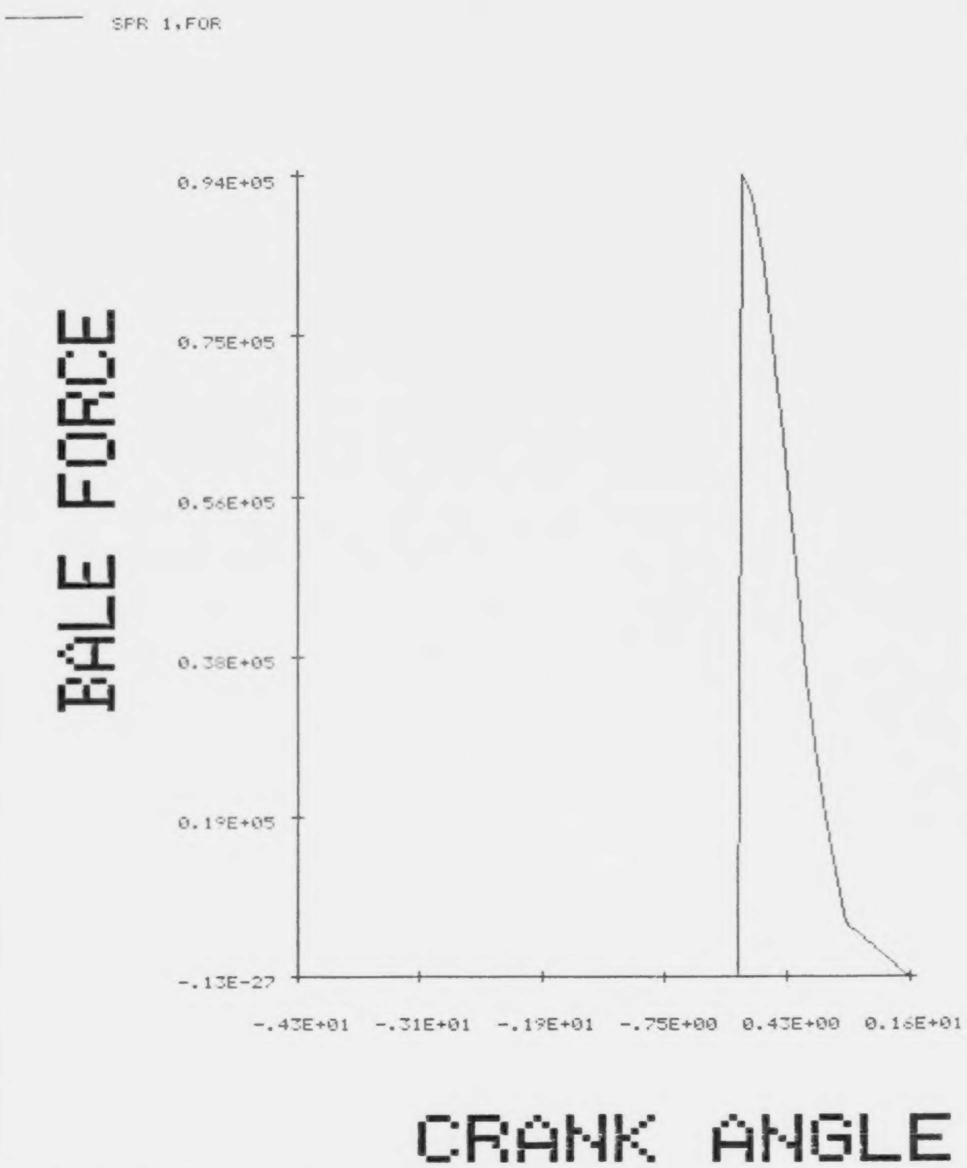


Figure 6. Bale Force on Plunger.

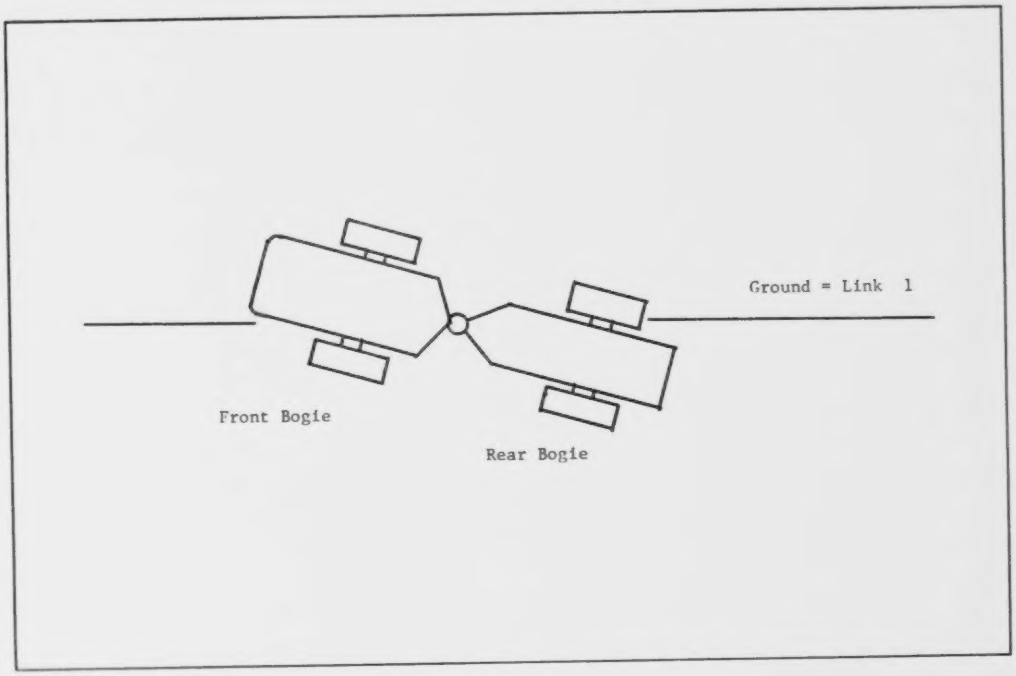


Figure 7. Articulated Tractor Model.

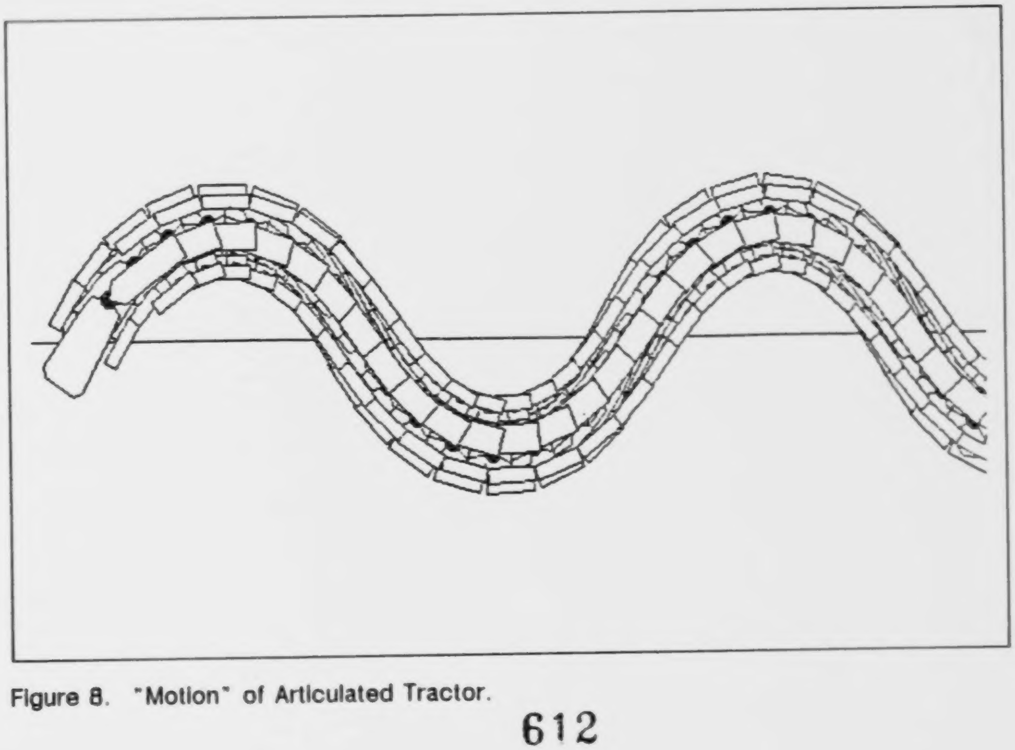


Figure 8. "Motion" of Articulated Tractor.

612

TEACHING NUMERICAL MATHEMATICS USING INTERACTIVE COMPUTER GRAPHICS

Presenter: **George F. Pinder** *Princeton University*

Coauthors: **George F. Pinder** *Princeton University*
Leonard J. Morse-Fortier *Princeton University*

INTRODUCTION:

A course in numerical methods provides a set of tools essential to modern engineering analysis. However, the amount and complexity of information contained in a comprehensive course makes both teaching and learning difficult in undergraduate education. To enhance an existing but underenrolled numerical methods course by eliminating the tedium of working many examples by hand, we created a new course as part of the IBM-AEP. As currently taught, the course includes five general topics, each presented using traditional lecture notes, student exercises, and an interactive computer program. The topics include polynomial approximation or curve fitting, numerical differentiation, numerical integration, solution of ordinary differential equations (ODE's), and solution of partial differential equations (PDE's). Lectures introduce the theory of each topic, include derivations of some of the relevant algorithms, and are run in parallel to computer precepts. During the computer precepts, we introduce the IBM-PC, discuss assignments, and try to provide time for starting those assignments. Student exercises include simple derivations and calculations, short programming assignments, and an interactive session with a large computer program. The interactive computer programs provide graphs of good resolution, high speed, and include enough flexibility to solve several student defined example problems.

PROJECT MOTIVATION:

A daunting amount of material must be learned to obtain a solid foundation in numerical methods. For an undergraduate, a two course elective sequence is often required to cover the material with sufficient breadth and thoroughness. Condensed into a single course, as is the case in our department, numerical methods was attracting very few students, and burdening those who did enroll with a large workload. This is because, in a thorough course, a student could expect to perform derivations, write computer programs for many of the relevant algorithms, and plot many curves in an effort to understand the material fully. To increase enrollment, we sought to make the course more attractive by integrating interactive computer graphics more completely. Our objective was to reduce the tedium of their tasks, provide a stimulating course, and thereby introduce the students to more material and cover that material more thoroughly.

CONCEPTUALIZATION:

In developing the course we encountered numerous ponderables. What should be the format of the course? How could the computer be integrated into the course? How much work should the computer do? How much should the students do? How does one assemble and manage a computer laboratory? What hardware would we use? How would it be configured? What software would form the basis of the course? How would we write it?

In addressing each of these and many other questions, we found some easier to deal with than others. The course was logically divisible into the five general subject areas introduced above. The computer is best at performing repetitive tasks, and so it was enlisted to solve a wide variety of sample problems, produce graphs of results, and be flexible enough to change. The overall course format evolved naturally from the need to balance lectures, student exercises, and interactive computer sessions.

Configuring and constructing the computer lab, producing the course software, and managing the lab were more difficult questions to address. A classroom was converted into the computer lab, and nineteen IBM-PC/XT 370's were installed. These computers included the ability to access Princeton's mainframe 370. The lab was intended to be public and general purpose, although our immediate concerns focussed on the logistics of the proposed course. A team of staff members and students worked together to design and write the course software within a framework of standardization, but the exact form of the software and the necessary standards necessarily evolved. Finally, the management of the lab presented several foreseeable problems, but many more unforeseeable problems were encountered and resolved when the course began.

IMPLEMENTATION:

While the above strategy answered the principal educational questions, many more logistical questions came up during the actual development of the course software. First, we had to design, develop, produce and debug large interactive programs within the environment of rapidly changing hardware and software support for the IBM–PC. To support interactive graphics, we needed to choose a specific graphics board and low level software driver that together could provide the speed and resolution we desired. Additionally, we had chosen to develop the software in FORTRAN, and the first available compilers were riddled with bugs and had architectural limitations which made large program development difficult. Eventually, IBM introduced Professional Fortran, and we were able to solve our problems. For graphics, we used the TECMAR Graphics Master board, and a privately developed driver that was several times faster than any commercially available, and we developed our own intermediate level graphics software, (this to conform to Princeton's mainframe graphics protocols.)

Problems encountered in managing the classroom seemed to arise out of the placement of personal computers in the public domain. We foresaw the problem of student privacy and security. Although each student was cautioned that any data stored on the computer's hard disk could be lost, we wanted to provide some protection. To this end, each student was assigned a subdirectory for their work, with a name that they chose, and which was hidden from the DOS DIRectory and TREE commands. This prevented others from copying or deleting their files. Unfortunately, we did not foresee the problem of naive users reformatting hard disks accidentally, but this happened on eleven machines in the first week the room was open. Also, system and course software was deleted by individuals working under CMS – the principal feature of the XT–370 – who required virtually all available local disk space.

We neither expected nor attempted to protect ourselves from sophisticated users, but we had to find the means to prevent careless and accidental damage. To this end, we "write" protected all essential and course software (running under DOS 2.1, no command was present for undoing this protection easily.) Also, we rewrote the format command to default and in fact to ONLY allow formatting of a floppy disk. Therefor, only a sophisticated user, with a fresh copy of DOS, could reformat a disk, or delete valuable software. These measures proved adequate, and, once implemented, saved hours of time.

The course software and curriculum were balanced to provide for student assignments requiring simple derivations, programming exercises, and interactive sessions with a topic program. Students were asked to keep a diary of their thoughts on each of several questions that they explored in these interactive sessions, and their reports provided us with valuable feedback on the computer programs and their effectiveness. Additionally, the thirty-five students who enrolled in the first course offering uncovered many interesting bugs in each program, and corrections were made throughout the course. Along with a diary for the interactive computer graphics assignments, students performed many of the more usual kinds of course exercises.

In studying polynomial approximation, for example, students programmed a cubic spline algorithm, but also studied LaGrange and Hermite polynomial fitting through the program CURVEFIT. The program allowed (x,y) data pairs to be entered manually, read from a file, or generated from a preprogrammed function. The nature of piecewise nth degree polynomial curvefitting was illustrated, and where data came from a preprogrammed function, plots of the error provided insight into the degree of precision, convergence, and numerical difficulties that can arise in fitting polynomials to data. In solving ODE's, programming assignments included solving a simple first order ODE by Galerkin's finite element method, and solving a nonlinear DE by iteration and finite differences. The program ODESOLVE illustrated several methods for solving ODE's including Euler, Euler Rhomberg, Runge Kutta, and Hanning. The program was designed to provide a sense of the degree of accuracy of each method, by comparing computed and known solutions.

Throughout the course, we tried to maintain a reasonable balance between working through programs with illustrative solutions and assigning the students exercises that explore the details of the methods.

RESULTS AND CONCLUSIONS:

The course, as measured through student evaluations and against our own objectives has been and continues to be a success. We covered the considerable amount of material required, student performance on exams, lab reports, and conventional assignments indicated reasonable levels of understanding, and enrollment increased by an order of magnitude.

The diary recording each topic program was a new experience for the students and forced them to qualify their understanding and describe their insights. Although at first bewildered by them, as the semester progressed, the students submitted better diaries and these provided us with a very good sense of how well people were learning and which exercises best illuminated the important ideas. Further, any lack of understanding or missing information became obvious in novel ways. For example, because the tedium of computation and plotting was handled by the computer, problems of conceptual misunderstanding were isolated from problems of careless numerical or algebraic mistakes which are common to other assignments.

THE USE OF INTERACTIVE THERMAL FINITE ELEMENTS IN AN UNDERGRADUATE HEAT TRANSFER COURSE

Presenter: A. F. Emery *University of Washington*

One of the common problems in the teaching of heat transfer is the lack of a method to visualize the interactions between the temperature and heat flux distributions. For example, radiative heat flux is proportional to the fourth power of the temperature. Thus small changes in temperature lead to large changes in heat flux. In reality, it is rare to have tem- perature defined boundary conditions; most are somewhere between prescribed temperature and prescribed heat flux. But large changes in boundary radiation heat fluxes imply only small changes in temperature; yet it is temperature which is sensed and best understood by the student. Consequently, the student has a difficult time visualizing how the temperature field changes as boundary conditions change and how it is related to the heat flux.

As an example, consider the two dimensional heat flow problem shown in Figure 1. The lower and left boundary are isothermal and the remaining two boundaries are adiabatic. The isotherms display the expected shape that the student has been shown in class and has derived from sketching the heat flow paths. Now consider the case where the central portion of the mesh is defined as having a very small value of conductivity. The distortion of the heat flux paths depends on the ratio of the conductivities. It is easy for the student to visualize the heat flux lines and to gain an understanding of the problem. Yet, when the student views a plot of the temperature, as illustrated in Figure 2, we find that the changes in the isotherms are, although marked to the eye of the expert, such that most students are unable to recognize that a central insulated area exists. In the past, numerical solutions, usually laboriously hand plotted, or machine plotted contours, using batch processing, have been used to assist the student.

Probably the best solution to this problem is an interactive program with a color graphics display of heat flux and temperatures. Using the program, the instructor can demonstrate in class or the student can investigate, at leisure, a wide range of properties and boundary conditions and gain an understanding of the complex relationship between heat flux and temperature. It is critical that the program and the graphics be fast; this is fairly easy to achieve. The program must be menu driven for flexibility, yet command structured to ensure that the student is adequately directed in the development of the solution; these are contradictory requirements.

To this aim, we have developed a family of finite element programs, with color graphic output of all relevant variables. Although the student can use either finite difference or finite element approaches, we have found the finite element method to be superior. The finite element methodology permits easy consideration of complex boundary conditions, rapid modification of input data, and is ideally suited to the 'What If' type of teaching which is necessary for undergraduates. Table 1 lists the features of finite element programs which are most important to teaching. One must be very careful in using numerical methods in the classroom. Anayltical solutions, although difficult to evaluate, usually give the student a complete understanding of the problem. Numerical methods, frequently give only a partial understanding and very sophisticated displays are necessary to complete the student's understanding. Fortunately, the finite element method is ideally suited to high speed, multi-color displays of temperature, heat flux, properties, stresses, and displacements.

TABLE 1

Why Finite Elements

a) Structured Algorithms
b) Modular Construction
c) Easily understood theory
d) Well suited to interactive programming
e) Represents a higher order language
f) Eliminates programming and I/O operations by the student
g) Provides an ideal graphical input and output

We have also found, that a well designed finite element program can be used by the graduate students in their research, although many differences have been found in the way that undergraduates and graduates respond to the program. Table 2 lists some of the different, but important, ways that the two classes of students view the programs.

TABLE 2

Needs of the Users
(The importance of each need is indicated by H(igh), M(edium), L(ow)
U(nimportant))

Undergraduates (or Faculty)	Feature	Graduates
M	Fast Processing	L
H	Bug-Free	L
H	Well Defined Program Flow	L
U	Modular Construction	H
H	Linear Problems	L
L	Non-linear Problems	H
L	Variable Properties	H
H	Steady State Problems	M
M	Transient Problems	H
U	User Defined Routines	H
H	Unambiguous Input	M
U	Command Driven	H
H	Menu Driven	U
H	On Line Help	M

Many of the differences are due to the entirely different nature of the level of expertise of the students and the type of problems considered. Graduate students are generally forgiving of bugs or errors since the ways of overcoming them are usually quickly known through word-of-mouth. Undergraduates, who in general are novices on the computer, have an extremely low tolerance level for program quirks or bugs. This also often applies to faculty. It has proven interesting that although color is a critical ingredient in helping the student understand the problem, simple line drawing of contours is sometimes to be preferred. The sudden breaks in the contours of Figure 2 emphasize the presence of a change in properties. The comparable color fringes usually are smoother and do not catch the students attention as easily. This lack of emphasis can be overcome by the ability to plot other variables, such as heat flux, stresses, and properties, which accentuate the changes in the computed results.

The different features of the programs are listed in Table 3.

TABLE 3

Features of the FLIP Programs
(All are Menu and Command Driven)

AUTOMATIC MESH GENERATION
Sub-parts (individually colored)
Moving, rotating, stretching
Replication, reflection
User chosen elements
Easy definition of material zones and boundary conditions
Automatic Bandwidth Reduction

STRESS ANALYSIS
Two Dimensional and Axisymmetric
Body Forces
Probability of Failure
Temperature Dependent Properties
Thermal Stresses
Anisotropic and Stratified Properties
Color Fringe display of all variables
Deformed mesh display

EQUIPMENT REQUIRED
IBM PC/AT, EGA with 128K piggyback
IBM Proprinter
IBM 6 pen plotter

THERMAL ANALYSIS
Steady State and Transient
Two Dimensional and Axisymmetric
Temperature Dependent Properties
Non-linear Boundary Conditions
Transient Sources
Color Fringe display of all variables
Aisotropic Properties

PLOTTING
Isometric, hidden line
Contours
Fringes
Can be viewed from any angle
Hard Copy output
Plotting of randomly space data

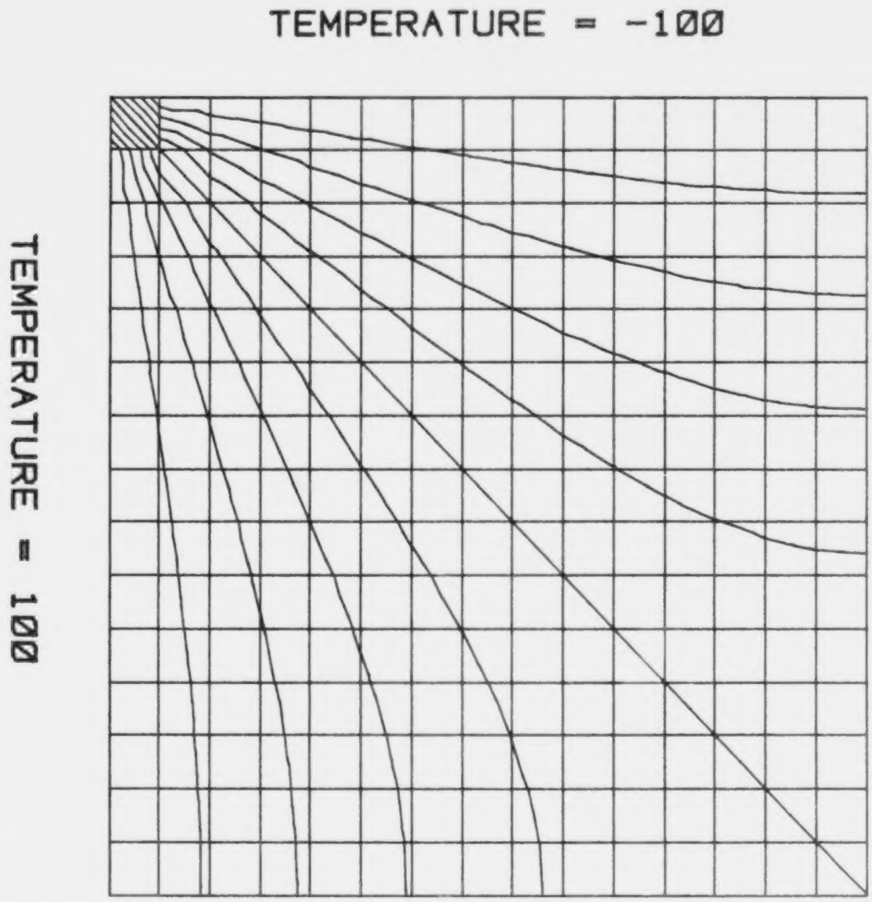


Figure 1. Temperature contours in a homogeneous plate.

622

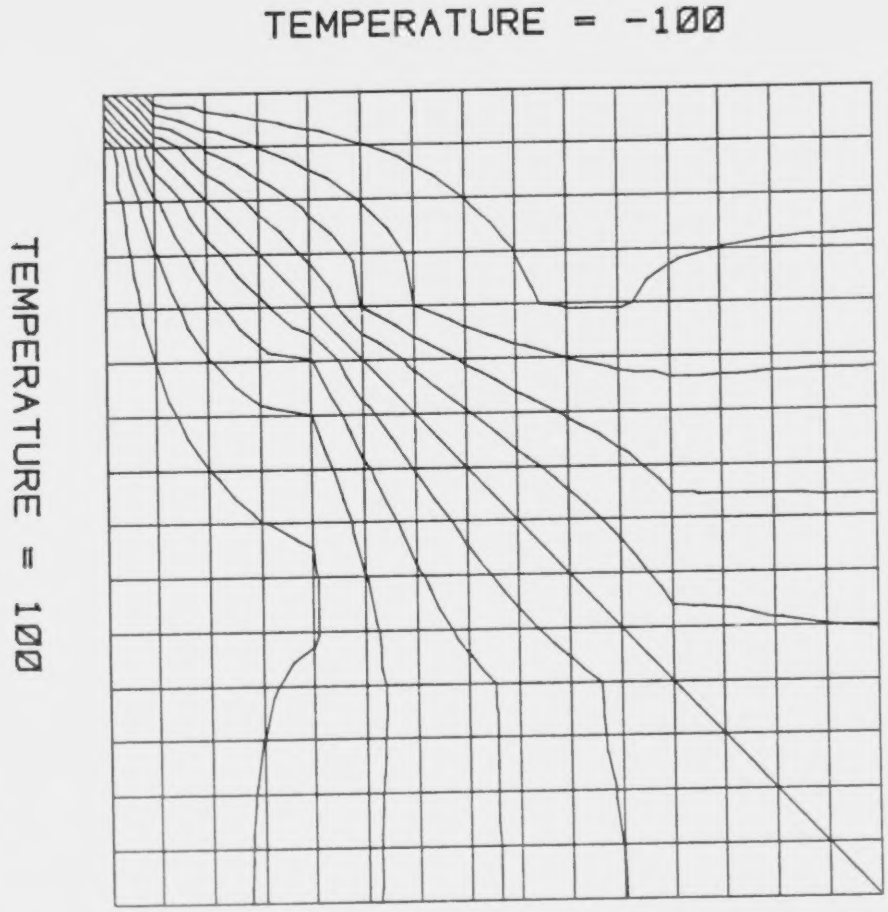


Figure 2. Temperature contours in a plate with an insulated center.

WHICH WAVE THEORY

Presenter: Rodney J. Sobey *University of California, Berkeley*

In much of the practice of coastal and ocean engineering, the question of which wave theory to use has been largely of academic interest. Linear (Stokes I) wave theory is accessible, familiar and easy to use. Consideration of shallow water and/or higher order wave theories has been substantially complicated by the relative inaccessibility of these theories. The literature is extensive and confusing, which has often led to the adoption of linear wave theory, whether or not it is appropriate.

Both graduate instruction and professional practice have suffered from this confusion. A recent paper by Sobey et al. (1986) has clarified much of this uncertainty and has resulted in a coordinated presentation of Stokes, Cnoidal and Fourier wave theory. This study has rationalized the application and microcomputer implementation of alternative wave theories but has identified a more deeply embedded problem, namely the identification of a valid mathematical solution as a valid physical solution. In much of the practical parameter range, mathematical predictions are available from a range of orders from Stokes, Cnoidal and Fourier theories. The differences are often considerable. In some cases, physically improbable solutions are quite clear but often the suspicious aspects can be overlooked if only a small part of the solution field is perused.

The choice of which wave theory requires perusal of the complete solution field for all field variables (horizontal and vertical velocity, horizontal and vertical acceleration and pressure) as well as the water surface profile. Computer graphics is the appropriate tool, specifically scalar contour plots of each field variable in a moving reference frame between adjacent wave crests.

MATHEMATICAL BACKGROUND

Progressive waves of permanent form are steady in a frame of reference moving at the phase speed C. Accordingly, it is convenient to adopt a steady or moving x, z reference frame (Figure 1) that is located at the sea bed and moves at speed C with the wave crest, rather than an unsteady or fixed reference frame. Assuming that the flow is incompressible and irrotational, the mathematical formulation may be presented in terms of the Euler equations, in terms of the velocity potential function or in terms of the stream function. Choosing the stream function ψ (x, z), the field equation representing mass and momentum conservation is the Laplace equation.

$$\frac{\partial^2\psi}{\partial x^2} + \frac{\partial^2\psi}{\partial z^2} = 0$$

where the velocity components (u,w) are ($\partial\psi/\partial z$, $-\partial\psi/\partial x$).
This field equation is subject to the following boundary conditions:
(i) Bottom boundary condition, representing no flow through the bed, is

$$\psi(x,0) = 0 \quad \text{at } z = 0$$

(ii) Kinematic free surface boundary conditions, representing no flow through the free surface, is

$$\psi(x,\eta) = -Q \quad \text{at } z = \eta(x)$$

where $\eta(x)$ is the free surface and $-Q$ is the constant volume flow rate per unit width under the steady wave.

(iii) Dynamic free surface boundary condition, representing constant atmospheric pressure on the free surface, is

$$1/2 \left[\left(\frac{\partial\psi}{\partial x}\right)^2 + \left(\frac{\partial\psi}{\partial z}\right)^2\right] + g\eta = R \quad \text{at } z = \eta(x)$$

where g is the gravitational acceleration and R is the Bernoulli constant.

(iv) Wave is periodic in space and time, which in the steady frame requires the wave profile to be symmetric in x about the crest.

The adopted solution methodology depends primarily on the dimensionless parameter $T(g/h)^{1/2}$, where T is the wave period and h is the mean water depth. For values less than about ten (deep water), a Stokes wave theory is usually adopted. For values greater than about ten (shallow water), a Cnoidal wave theory is usually adopted. Alternatively, a Fourier approximation theory may be used, regardless of the value of $T(g/h)^{1/2}$. The theoretical development of rational and consistent theories from first to fifth order for Stokes and Cnoidal waves, and from first to (theoretically) any higher order for Fourier approximation waves have much in common. Sobey et al. (1986) have emphasized the common aspects in establishing a coordinated presentation of these theories, closely following a sequence of papers by Fenton (1979, 1981, 1985).

The detail of these wave theories is crucial but this is not the context in which to dwell on such detail. It is sufficient to observe that each theory is analytically complicated, becoming also numerically complicated at higher orders and for all orders with Fourier wave theory. Sobey et al. (1986) have presented tabular summaries of the predictive equations for Stokes, Cnoidal and Fourier theories.

MEASURES OF VALIDITY

The ultimate test of the validity of any theory is a comparison of theoretical predictions with laboratory and field measurements. This was the approach adopted by Le Mehaute, Divoky and Lin (1968). Their detailed conclusions in 1968 however may no longer be appropriate. There have been significant refinements in Stokes, Cnoidal and Fourier wave theory in the last decade. The well-known Skjelbreia and Hendrickson (1962) Stokes V theory has been shown to be incorrect at fifth order. The impact of a coflowing current on higher order wave theories has been identified by Fenton (1985). This is especially relevant to the comparison of theoretical predictions with laboratory flume measurements; a condition of zero Stokes drift must be imposed on the theory, just as it is in the flume. Fenton has illustrated the consequences, showing that some apparent disagreements were largely an error in the theoretical prediction. There is certainly a case for repeating these comparisons.

An alternative measure of validity is a comparison of theoretical predictions with the imposed free surface boundary conditions, the approach adopted by Dean (1970). This may seem unnecessarily selective in considering only Equations 3 and 4 but it is quite appropriate. The analytical foundation of Stokes, Cnoidal and Fourier wave theories all ensure that the field equation (Equation 1) and the bottom boundary condition (Equation 2) are satisfied identically. Dean adopted the mean square errors in the kinematic and dynamic free surface boundary conditions as measures of validity. His results are predictable. Stokes waves do best in deep water, Cnoidal waves do best in shallow water, the transition region between Stokes and Cnoidal waves is somewhat uncertain, and Fourier waves do very well throughout except in very shallow water.

The restriction of attention to the free surface however may not provide a sufficiently comprehensive evaluation to identify potentially spurious solutions.

The complexity of the analytical detail almost guarantees that potential problems are deeply embedded in the predictive equations. To establish without question the appropriateness of a wave prediction is perhaps an unreasonable goal, but to establish that there are no questionable features of the solution field is a reasonable objective. In our experience, it can prove most revealing.

THE NATURE OF THE PROBLEM

One of the difficulties in application of the Stokes and Cnoidal theories has been the limited range of validity of the respective theories. Mathematically correct but physically incorrect solutions can often be computed. Unless these solutions are carefully studied before further use, serious errors may result. Figure 2 includes illustrations of spurious solutions that can easily arise. Three fifth order predictions are superimposed, Stokes V, Cnoidal V and Fourier V, for the same wave. The wave height is 8.55 m, the water depth is 15.61 m and the wave period is 10 s, corresponding to Dean Case 6C (Dean 1974) and representing about three-quarters of breaking wave conditions in transitional water. The predicted wave lengths are very close, 121.3 m, 121.6 m and 122.3 m respectively, but there is considerable variation in the predicted wave profiles in Figure 2a. The Stokes V prediction has four reversals of curvature. It is nonetheless a perfectly correct mathematical solution of the gravity wave equations at fifth order, as the wave height is defined (Figure 1) as the elevation difference between the crest at $x = 0$ and the trough at $x = L/2$. However, this is clearly not a correct physical solution. Fenton (1985) has shown that the effective Stokes expansion parameter in shallow water is not ϵ but $\epsilon/(kh)^3$. Here ϵ is 0.57 and is perhaps sufficiently small but $\epsilon/(kh)^3$ is 1.20 and not sufficiently small for the Stokes theory to remain appropriate. The Cnoidal V prediction is also a perfectly correct mathematical solution at fifth order and the wave profile prediction in Figure 2a is perhaps plausible. The horizontal velocity profile prediction under the wave crest shown in Figure 2c now serves to identify the Cnoidal V prediction as physically invalid. The magnitude is very large and there are two curvature reversals with a velocity minimum at about two-thirds of the depth. The expansion parameter for horizontal velocity, ϵ/m is 0.73 here, much too large for the truncated series to give reasonable predictions. Note, however, that the Stokes V prediction of horizontal velocity is much more plausible. Further confirmation of the physical invalidity of the Cnoidal V solution is provided by Figure 2b, showing horizontal acceleration at the water surface. The extreme magnitude of the acceleration under the crest is most improbable. There are problems also in the Fourier V prediction to which the discussion will return but it is quite close to the mathematically and physically correct solution. It is important to appreciate that the mere achievement of a mathematical prediction from a wave theory does not guarantee its physical validity. Figure 2 is a good illustration of the need to carefully screen the complete solution. The Fourier XVIII solution is also included on Figure 2 to give a datum for comparison; this solution is mathematically and physically correct.

Fourier approximation wave theory is a numerical wave theory that seeks to substitute computational effort for the limited range of applicability of analytical wave theories. The wave field (specifically the stream function) is represented as a Fourier series whose coefficients are computed numerically to collectively satisfy the free surface boundary conditions, not individually at each order as for analytical waves theories. Fourier wave theory has nominally uniform validity from deep to shallow water. Problems such as depicted in Figure 2 are largely avoided, but at the expense of a significant increase in the required computational effort. It is a computational effort, however, that is reasonably accommodated on a microcomputer.

Microcomputer implementation of Fourier wave theory, however, does confront a potential difficulty of the theory that may be overlooked in the use of tabulations (Dean 1974). This is the Gibbs' phenomenon, a consequence of Fourier interpolation between computational nodes. A Fourier theory of order N has N + 1 computational nodes at $x_1 = iL/2N$, $i = 0, 1, \ldots N$. Vertical profiles at these x_1 positions and wave or horizontal profiles using only these x_1 retain the precision of the theory. For a Fourier V theory, these positions are at 0%, 10%, 20%, 30%, 40%, and 50% of the wavelength respectively. Wave and horizontal profiles using additional points as well as vertical profiles at other positions are rationally established from the predictive equations (Fenton 1980, Sobey et al. 1986), as the Fourier coefficients are known. In each case this involves Fourier interpolation and the inevitable Gibbs' phenomenon, the appearance of oscillations in the profile. These oscillations are mostly small but they can lead to lack of confidence in the Fourier predictions if their source is not understood. The apparent problem can of course be avoided by coordinating the output profile locations with the order of the theory. The Fourier V prediction in Figure 2 illustrates this problem. The order is five but the profiles are based on fifty uniform steps in a half-wavelength. There is a small oscillation in the Fourier V wave profile but there is of course no oscillation in the horizontal velocity profile as its location at the crest corresponds to a computational node. This oscillation is much more pronounced in Figure 3. A Fourier XVIII prediction presented at eighteen uniform steps in a half-wavelength does not exhibit any wave profile oscillations. The Gibbs' phenomenon can of course not be avoided by numerical tabulation of the predictions; there are oscillations, for example in the tabulated predictions for wave profile for Dean Cases 1D and 4D (Dean 1974), despite the relatively coarse resolution.

IS A MATHEMATICAL SOLUTION A PHYSICALLY VALID SOLUTION?

It is apparent from the above discussion that the identification of a physically invalid solution may not be straightforward. The initial step in Stokes and Cnoidal wave theory is the solution of the dispersion relationship

$$\frac{L}{T} \quad \bar{u} + C_E$$

where T is the wave period and C_E the coflowing Eulerian current. The wavelength L and the mean fluid speed $\bar{u}$ depend on the wave theory, the order of the theory and on the wave number k for Stokes waves or the Jacobian parameter m for Cnoidal waves. In all cases, it is an implicit algebraic equation, whose complexity increases with order. In the more extreme cases, physical invalidity is apparent at this stage. Either the equation does not converge or it converges to a solution that is immediately identified as physically impossible; a negative value of k or a Jacobian m outside the range zero to one are common examples.

In less extreme cases, mathematically correct solutions are achieved that cannot immediately be rejected. Figures 2 and 3 provide illustrations of such situations. The problems can be quite subtle and not immediately apparent. Parts of the predicted solution may seem most plausible (e.g. water surface profile for Cnoidal V, horizontal velocity profile under crest for Stokes V) but other parts are equally implausible (e.g. water surface profile for Stokes V, horizontal velocity profile under crest for Cnoidal V). Fourier wave theory provides its own list of potential problems. Oscillations associated with the Gibbs' phenomenon have already been mentioned, and are illustrated in Figure 2 and especially Figure 3. The multi-dimensional implicit equation formulation of Fourier wave theory leads to another reasonably common problem–convergence to another solution than the solution specifically sought. This is a major problem in shallow water where the water surface profile around the crest is quite sharp and reasonably high order solutions become necessary. Without considerable care, convergence will be directed towards a higher harmonic, commonly the third. This is an irritant but not an error. It is a physically valid wave that correctly conforms to the mathematical problem formulation, but it is not the solution sought. This problem is immediately identified from a plot of the water surface profile.

From the variety of problems encountered in evaluating a mathematical solution, it is apparent that detailed scrutiny of the predicted solution field is essential before the predictions are used in further analysis. Higher order wave theories often provide the input or forcing to sophisticated computations, such as a dynamic structural analysis of an offshore platform. Serious errors in the wave theory prediction may become hidden in the complexity and perhaps volume of the subsequent analysis and they might not be detected. A detailed perusal would involve all dependent variables of the wave field, specifically the water surface profile, horizontal and vertical velocities, horizontal and vertical accelerations and the pressure. Additional to this list would be the temporal and convective acceleration components; the nature of these components and the predictive equations are sufficiently different to warrant separate consideration. Separate consideration should also be given to the dynamic pressure contribution to the total pressure. With the exception of the wave profile, all of these dependent variables are functions of x and z in the steady reference frame.

EVALUATION OF SOLUTION FIELDS

The appropriate procedure for rapid perusal of scalar field variables is contour mapping in the x,z plane, within the region bounded in x by the crest and the trough and in z by the bed and the free surface. This approach has been adopted, utilizing a contour algorithm for a regular grid developed by Heap (1974). This algorithm is based on a regular grid of contour heights. The adopted computational spacing was $\Delta x = L/100$ and $\Delta z = h/20$. This algorithm does not cope well with the free surface, above which the contour height is set to zero.

The horizontal velocity component has been chosen for presentation in Figures 4 through 9. Figures 4 through 8 correspond with the Dean Case 6C solution presented partially in Figure 2. The linear wave theory or Stokes I solution in

Figure 4 is included for perspective -- this is the classical lowest order solution. The Stokes V solution (Figure 5) shows the sharpening of the crest and flattening of the trough profile that is characteristic of higher order solutions. The changes from Figure 4 are otherwise relatively small and there is no indication from this horizontal velocity field that the solution is in any way suspect, except for the curvature reversals in the water surface profile to which reference has already been made.

The dominant feature of the Cnoidal I solution (Figure 6) is the uniform velocity profiles with depth. This is a characteristic feature of Cnoidal I waves. The velocity magnitudes are reasonably consistent with Figures 4 and 5, and the water surface profile is sharpened at the crest and flattened at the trough, as expected. There is no evidence also to dispute the physical validity of this solution. The Cnoidal V (Figure 7) solution field is immediately identified as a physically improbable solution, especially the extreme magnitudes of the horizontal velocity near the crest; such velocities, exceeding 150 m/s, are just not physically realizable. Note however that the predicted water surface profile is most plausible. The Fourier XVIII solution field (Figure 8) is also included to provide a higher-order perspective. This solution is both physically and mathematically acceptable.

The final solution field (Figure 9) corresponds to the Fourier V solution in Figure 3. Note the influence of the free surface oscillations on the velocity field. The water is also quite shallow and the solution is correctly predicting for an almost uniform vertical velocity distribution, despite the free surface difficulties.

CONCLUSIONS

The analytical complexities of higher-order Stokes, Cnoidal and Fourier wave theories can lead to some rather spurious predictions. This is especially true in parameter ranges where these theories are being pushed to and perhaps beyond their limits of physical validity. Unfortunately in many of these cases, a mathematical solution is still achievable.

The question of whether a mathematical solution can be identified as an appropriate physical solution has been shown to involve a number of perhaps unexpected subtleties. It is necessary to peruse the complete solution field for all dependent variables to confirm that a mathematical solution is indeed physically acceptable. It is demonstrated that contour mapping and computer graphics are the appropriate tools for this

ACKNOWLEDGEMENT

The research leading to this paper made extensive use of microcomputer hardware and software provided to the Department of Civil Engineering through the IBM DACE Grant to the University of California, Berkeley.

REFERENCES

Dean, R. G., "Relative validity of water wave theories, **Journal of Waterways, Harbors and Coastal Engineering, ASCE**, Vol. 96, 1970, pp. 105–119.

Dean, R.G., "Evaluation and Development of Water Wave Theories for Engineering Application," Coastal Engineering Research Center, Special Report No. 1, 1974, 2 volumes.

Fenton, J.D., "A High-Order Cnoidal Wave Theory," **Journal of Fluid Mechanics**, Vol. 94, 1979, pp. 129–161.

Fenton, J.D., "A Fifth-Order Stokes Theory for Steady Waves," **Journal of Waterway, Port, Coastal and Ocean Engineering, ASCE**, Vol. 111, 1985, pp. 216–234.

Fenton, J.D., "The Numerical Solution of Steady Water Wave Problems," **Computers and Geosciences**, Vol. 12, 1986, to appear.

Heap, B.R., "Two Fortran Contouring Algorithms," National Physical Laboratory, England, Report NAC 47,1974.

Le Mehaute, B., Divoky, D., and Lin, A., "Shallow water waves: A comparison of theories and experiments," Procs., 11th Int. Conf. Coastal Engineering, London, 1968, pp. 87–107.

Rienecker, M.M. and Fenton, J.D., "A Fourier Approximation Method for Steady Water Waves," **Journal of Fluid Mechanics**, Vol. 104, 1981, pp. 119–137.

Skjelbreia, L. and Hendrickson, J., "Fifth Order Gravity Wave Theory," Procs., 7th Int. Conf. Coastal Engineering, The Hague, ASCE, 1962, pp. 184–196.

Sobey, R.J., Goodwin, P., Thieke, R.J. and Westberg, R.J., Jr., "Application of Stokes, Cnoidal and fourier Wave Theories," manuscript under review, 1986.

633

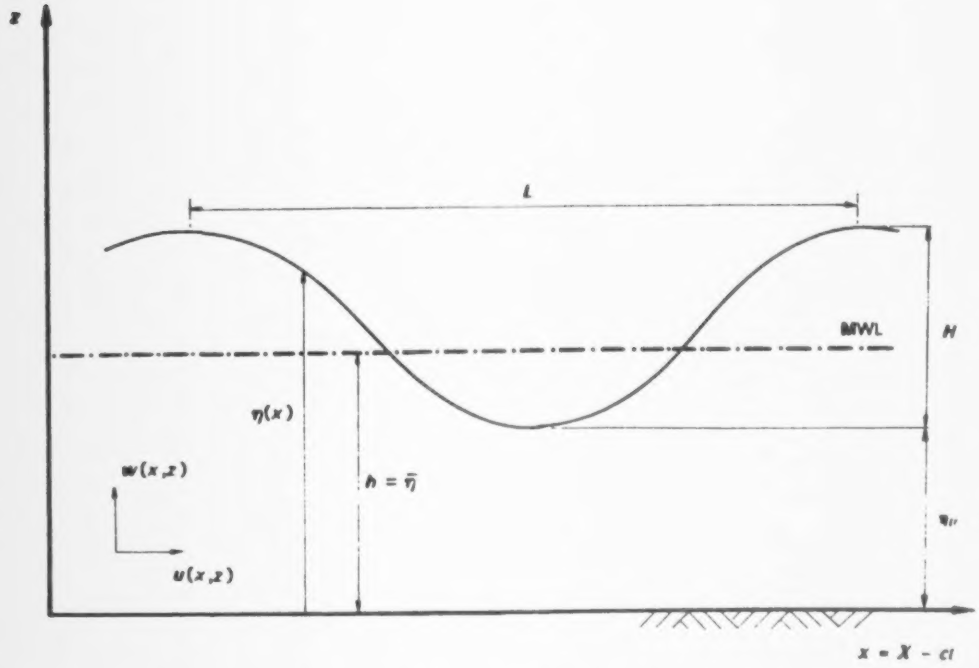


Figure 1. Definition of variables in a steady reference frame.

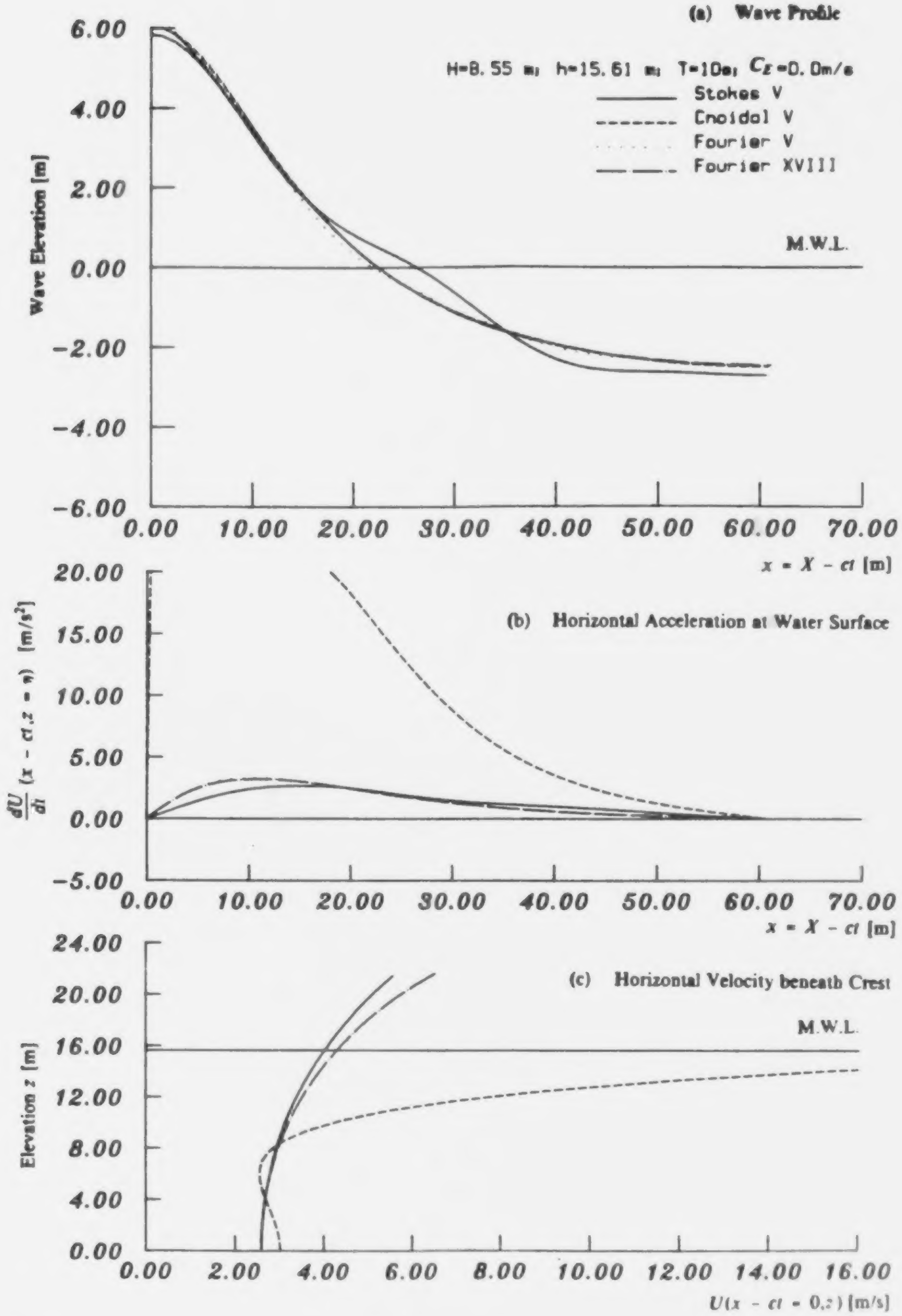


Figure 2. Stokes V, Cnoidal V and Fourier V Predictions for Dean Case 6C.

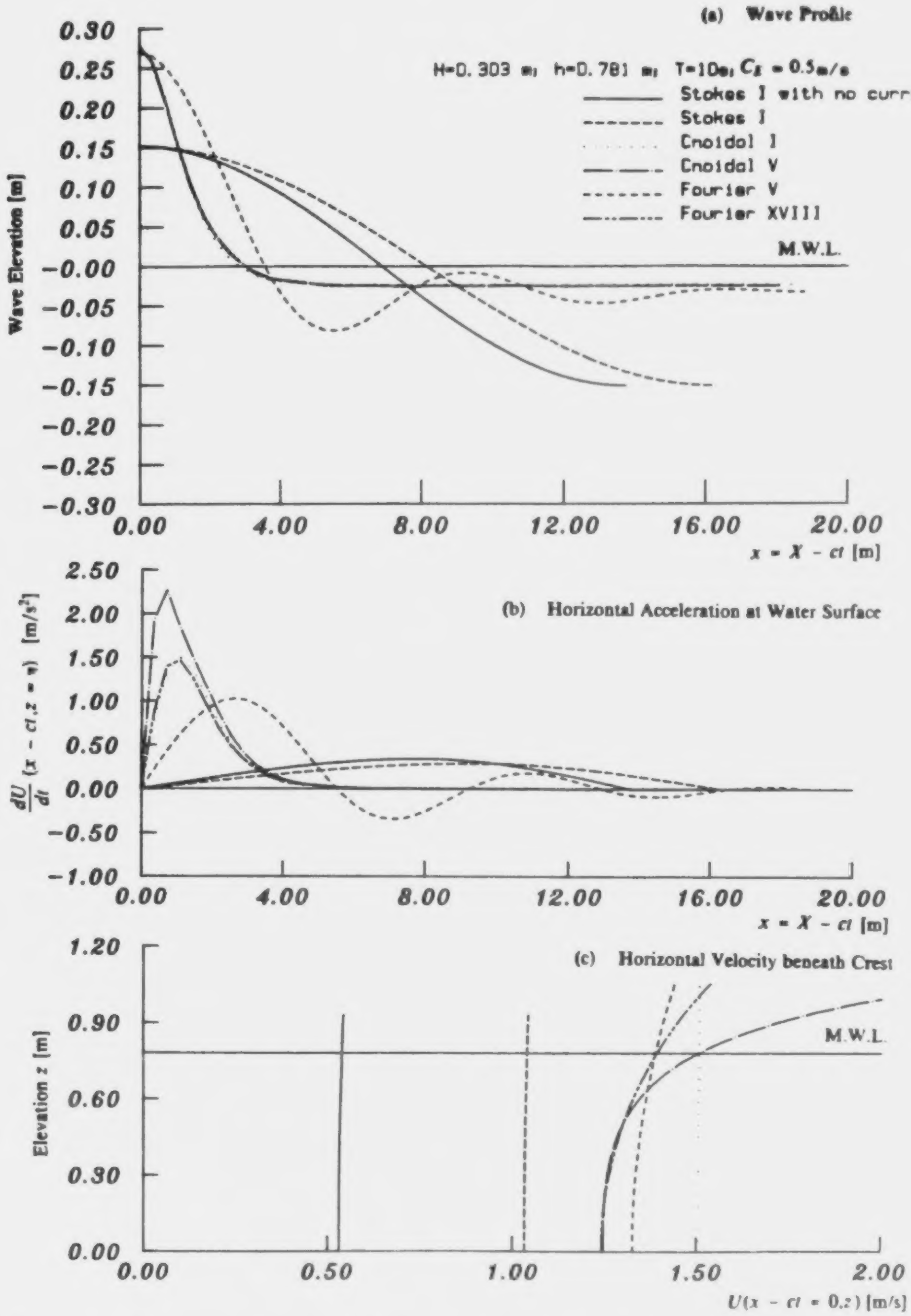


Figure 3. Stokes, Cnoidal and Fourier Predictions for Dean Case 2B with 0.5 m/s Current.

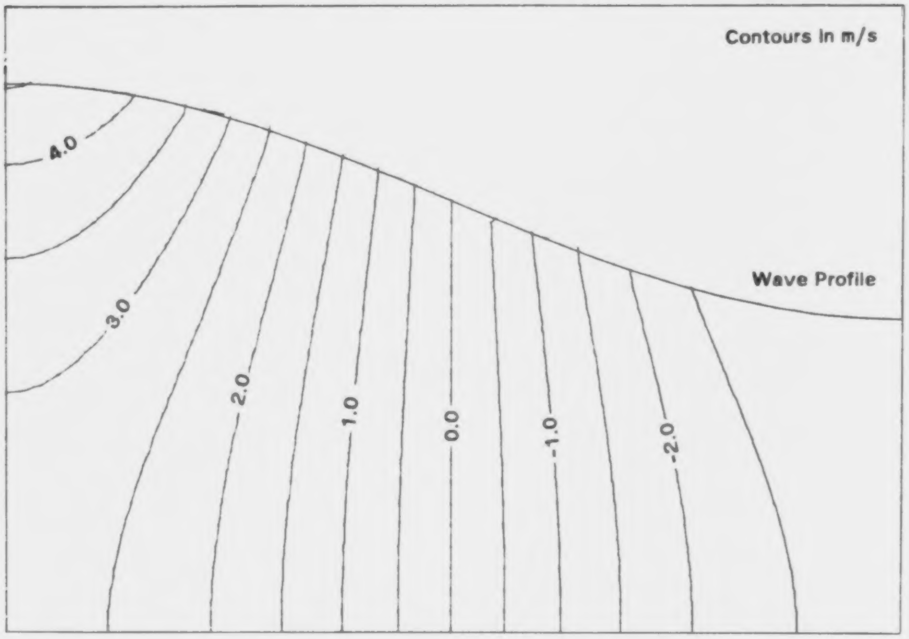


Figure 4. Stokes I Horizontal Velocity Field for Dean Case 6C.

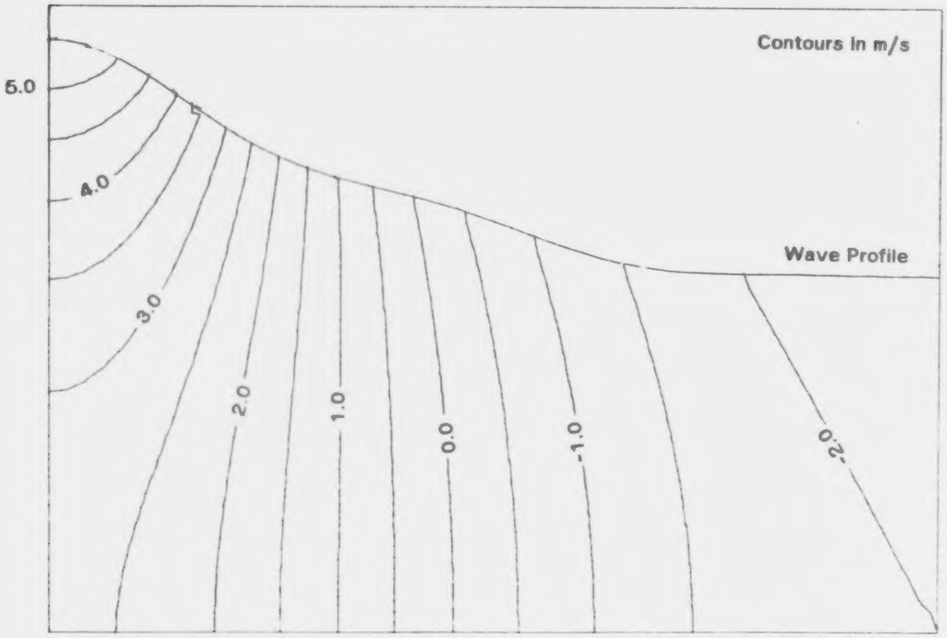


Figure 5. Stokes V Horizontal Velocity Field for Dean Case 6C.

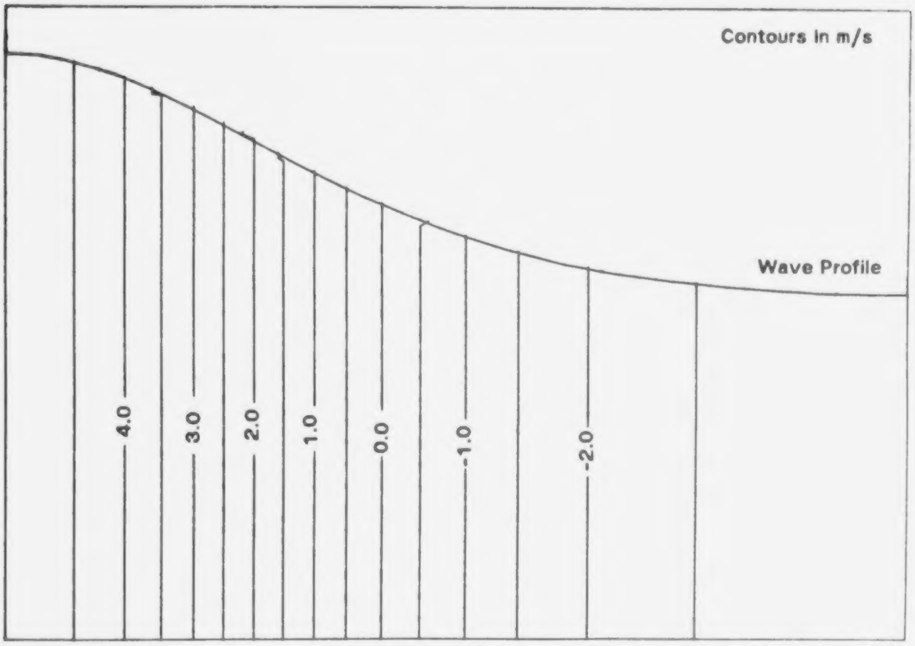


Figure 6. Cnoidal I Horizontal Velocity Field for Dean Case 6C.

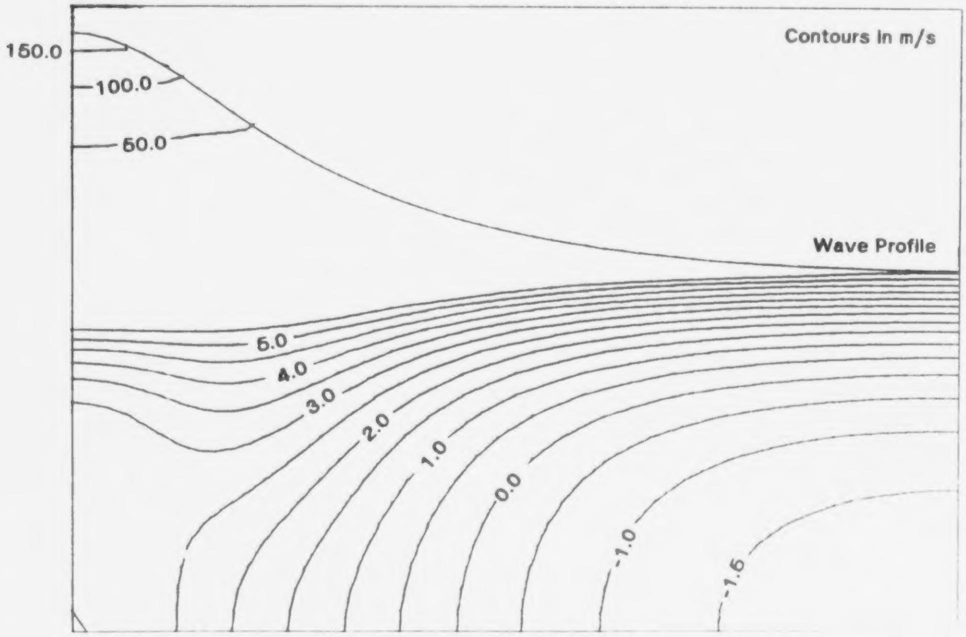


Figure 7. Cnoidal V Horizontal Velocity Field for Dean Case 6C.

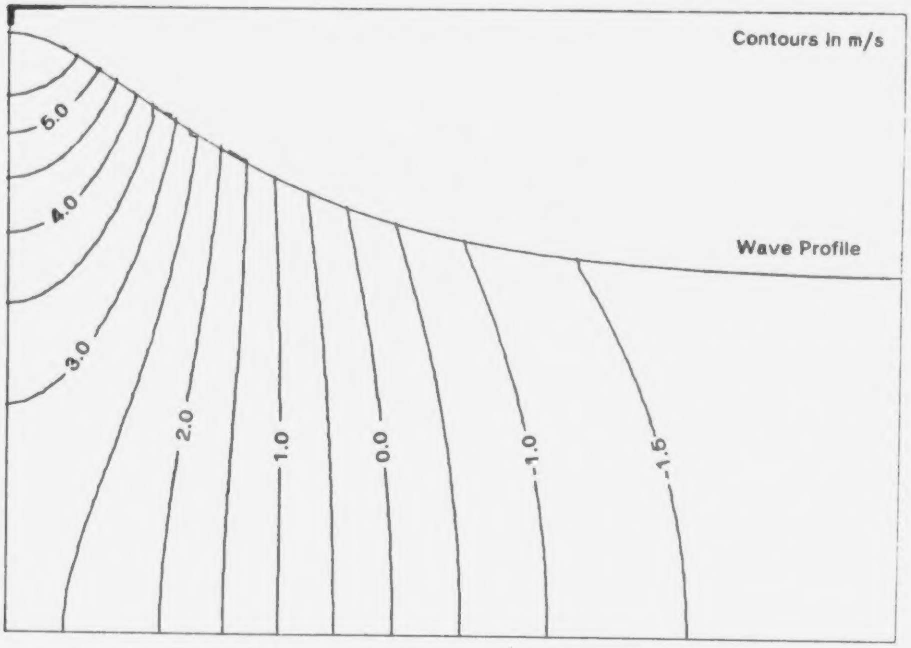


Figure 8. Fourier XVIII Horizontal Velocity Field for Dean Case 6C.

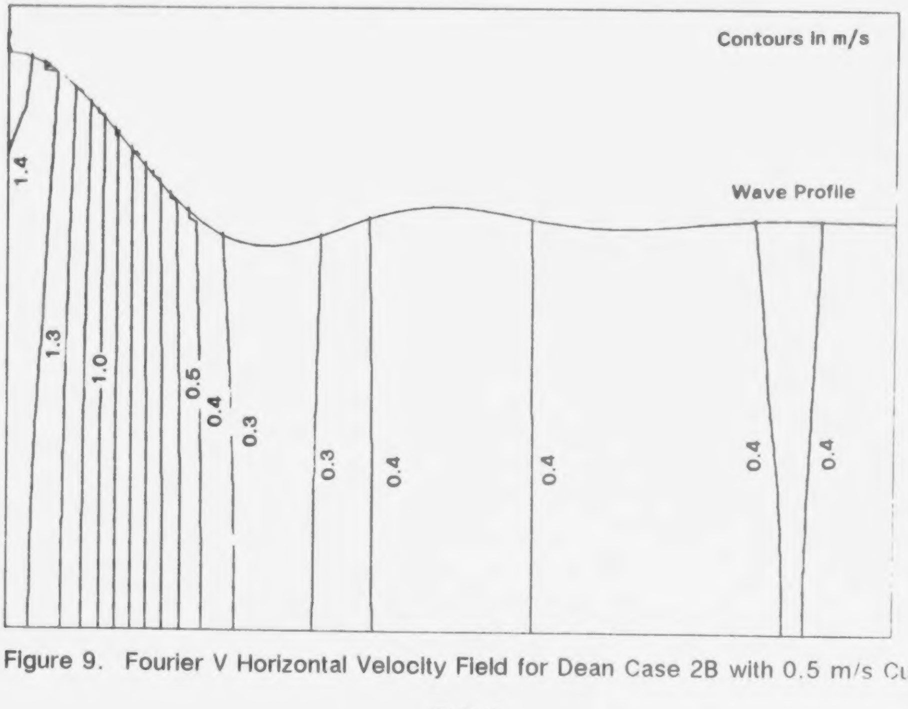


Figure 9. Fourier V Horizontal Velocity Field for Dean Case 2B with 0.5 m/s Current.

INDEX BY PRESENTER / AUTHOR, VOLUME V

Presenter / Author **Page**

Beckman W. A. ... 95

Chieng, Wei–Hua ... 36

Coddington, Richard C. ... 179

Compton, Richard ... 73

Dickinson, Bradley W. ... 91

Emery, A. F. ... 196

Heersink, Roland E. ... 104

Hoeltzel, David A. ... 36

Klein, S. A. ... 95

Lewin, D. R. ... 155

Loucks, Daniel P. ... 168

Maher, Mary Lou ... 141

Milne, Murray ... 2

Monismith, Carl L. ... 57

Morari, Manfred ... 104

Moromisato, G. P. ... 79

Morse–Fortier, Leonard J. ... 191

Nollert, M. U. ... 79

Olbricht, W. L. ... 79

Oppenheim, Irving ... 141

Pinder, George F. ... 191

INDEX BY PRESENTER / AUTHOR, VOLUME V

Presenter/Author	Page
Reeves, Anthony P.	25
Rutledge, David	73
Skjellum, Anthony	104
Sobey, Rodney J.	215
Soma, Mani	16
Sousa, Jorge B.	57
Williams, Wyman	73
Woodbury, Robert F.	202
Zhang, Weiguang	141

INDEX BY UNIVERSITY, VOLUME V

University **Page**

California Institute of Technology 73, 104, 155

Carnegie–Mellon University 141, 202

Columbia University 36

Cornell University 25, 79, 168

Princeton University 91, 191

University of California, Berkeley 57, 215

University of California, Los Angeles 2

University of Illinois at Champaign–Urbana 179

University of Washington 16, 196

University of Wisconsin 95

INDEX BY PRESENTER/AUTHOR, VOLUMES I–V

Presenter/Author Vol Pg

Alexander, Kirk .... IV 152
Anderson, Gary F. .... III 114
Atkinson, Daniel E. .... III 81
Barkley, David S. .... III 81
Beauchamp, James .... II 60
Becker, Donald A. .... II 91
Beckman W. A. .... V 95
Billman, E. .... IV 38
Bogom, Stuart .... IV 65
Boh, Larry E. .... III 77
Bolger, Michael B. .... III 2
Bowes, John E. .... IV 2
Braca, Michael J. .... IV 92
Bradley, Joan .... III 85
Braune, R. J. .... IV 38
Brauth, Steven E. .... III 16
Causey, Robert L. .... II 64
Chieng, Wei-Hua .... V 36
Churchill, Lynn .... IV 111
Clarke, Steven G. .... III 81
Coddington, Richard C. .... V 179
Cohen, Mark R. .... II 101
Comeaux, Michelle .... IV 170
Compton, Richard .... V 73
Danielson, Wayne .... IV 102

INDEX BY PRESENTER/AUTHOR, VOLUMES I–V

Presenter/Author	Vol	Pg
Dannenberg, Roger B.	II	109
Davis, T.	IV	38
Dickinson, Bradley W.	V	91
Emery, A. F.	V	196
Entwistle, W. C.	IV	38
Farber, G.	III	120
Flach, J. M.	IV	38
Fliess, Kenneth H.	I	52
Gardy, Martin	III	93
Gillman, Clifford	IV	170
Godley, Paul	III	85
Godley, Susan	III	85
Golde, Hellmut	IV	18
Goldman, Steven M.	I	2
Guenin, Mark A.	IV	11
Gutmann, Myron P.	I	52
Hagamen, W. D.	III	93
Haltiwanger, John	I	108
Heersink, Roland E.	V	104
Helm, Patricia Burton	IV	54
Hill, John W.	II	45
Hoeltzel, David A.	V	36
Holmes, Randy D.	I	45
Horan, Fred	II	11
Horowitz, Ellis	IV	98

INDEX BY PRESENTER/AUTHOR, VOLUMES I–V

Presenter/Author	Vol	Pg
Hunter, Scott	II	11
Kane, D. J.	III	124
Kidd, Kenneth K.	I	45
Klein, S. A.	V	95
Kohler, B.	III	120
Kokkinos, Chris	II	11
Kroch, Eugene	I	85
Krull, James	III	85
Kruse, U. E.	III	124
Lachenbruch, Peter A.	I	62
Lamb, Sandra I.	III	81
Lee, James L.	I	13
Lewin, D. R.	V	155
Lidell, Gail	II	11
Lintern, G.	IV	38
Lomax, Anthony	III	109
Loucks, Daniel P.	V	168
Ludden, Thomas	III	85
Mackenzie, Arch	II	11
Maher, Mary Lou	V	141
Mahoney, M.	IV	60
Mane, A.	IV	38
Martin, Robert L.	II	83
McGraw, Deborah	II	32
Miller, David W.	I	97

INDEX BY PRESENTER/AUTHOR, VOLUMES I–V

Presenter/Author	Vol	Pg
Miller, Kevin F.	I	34
Milne, Murray	V	2
Mintz, Norman	IV	23
Monismith, Carl L.	V	57
Moore, John	III	85
Morari, Manfred	V	104
Moromisato, G. P.	V	79
Morse–Fortier, Leonard J.	V	191
Mulligan, Ed	III	72
Niesz, Anthony J.	II	48
Niklas, Karl J.	III	65
Noblitt, J. S.	II	36
Nollert, M. U.	V	79
Olbricht, W. L.	V	79
Oppenheim, Irving	V	141
Oppenheimer, Joe	IV	83
Owen, David I.	II	2
Paperno, Slava	II	32
Pet, W. J. A.	II	36
Pinder, George F.	V	191
Pinkston, Russell	II	78
Pitterle, Michael E.	III	77
Rae, Douglas W.	I	66
Rees, Douglas C.	III	81
Reeves, Anthony P.	V	25

INDEX BY PRESENTER/AUTHOR, VOLUMES I–V

Presenter/Author	Vol	Pg
Robert, Jean-Christophe	V	156
Robinson, Bruce H.	III	114
Rohn, Peter H.	I	112
Rosenstone, Steven J.	I	66
Rutledge, David	V	73
Sagrave, Rosalie	III	85
Schmitt, Gerhard	IV	156
Schultz, Larry W.	III	77
Schwartz, William	III	147
Shapiro, Matthew D.	I	40
Sherwood, Bruce Arne	IV	29
Sherwood, Judith N.	IV	29
Siler, Randy	IV	111
Skjellum, Anthony	V	104
Smith, J. H.	III	124
Smith, Ronald	III	142
Sobey, Rodney J.	V	215
Sol, D. F.	II	36
Soma, Mani	V	16
Sousa, Jorge B.	V	57
Steinfel, J. I.	III	120
Stenstrom, Michael K.	IV	76
Stixrude, L.	III	131
Stokes, A. F.	IV	38
Stolp, Chandler	I	23

INDEX BY PRESENTER/AUTHOR, VOLUMES I–V

Presenter/Author	Vol	Pg
Sutter, Carl	II	11
Taylor, Edwin F.	III	41
Tettleman, Bruce	IV	73
van Dam, Andries	IV	92
VanderVen, Ned S.	III	29
Walker, Russell C.	III	10
Weller, M. H.	IV	38
Wenk, H. R.	III	131
Wickens, C. D.	IV	38
Wiederholt, Joseph B.	III	77
Wilder, H. A.	III	120
Williams, David R.	I	28
Williams, Wyman	V	73
Winer, Mark	IV	83
Woodbury, Robert F.	V	202
Zhang, Weiguang	V	141

INDEX BY TITLE, VOLUMES I–V

	Vol	Pg
A "Language Lab" for Architectural Design	II	11
A Computerized Sumerian Dictionary of the Third Millennium	II	2
A Generalized Videotex Laboratory	IV	2
A Microcomputer Tool to Design Energy Conserving Buildings	V	2
A PC–Based VLSI Design System for Education	V	16
A Practical Authoring System/CBI with a Video Digitizer	IV	11
An Administrator's View of AEP Program Operation and Evaluation	IV	18
An Authoring System as a Tool for Foreign Language Learning	II	32
An Instructional Facility for Computer Vision/Image Processing	V	25
An Intelligent Chemical Stencil for 3D Molecular Coordinates	III	2
An Interactive System Approach to Near Optimal Design	V	36
Beyond AEP	IV	23
Bilingual Word Processing: A New Tool for Learning	II	36
Calculus Software From the PC to the RT	III	10
Closed–Loop Dynamic Testing Control Using an IBM PC/AT	V	57
CMU Tutor:An Integrated Programming Environment for Workstations	IV	29
Computer Aided Instruction in Neuroanatomy	III	16
Computer Aided System Analysis and Design	V	91
Computer Assisted Lectures	I	2
Computer Augmentation of Flight Simulator Training	IV	38
Computer Based Instructional Materials in Counseling Psychology	I	13
Computer Graphics Flow Visualization	V	79
Computer Graphics for an Intermediate–Level Optics Course	III	29
Computer Graphics Utilities in Special Relativity	III	41
Computer Simulations of Plant Morphology and Evolution	III	65

INDEX BY TITLE, VOLUMES I–V

Vol Pg

Computer Text Analysis/Database Management, Musicological Study .... II 45

Computer–Aided Design of Microwave Integrated Circuits .... V 73

Computerized Instruction in Engineering Thermodynamics ....V 95

Computers in Residence: Social and Managerial Issues ....IV 54

Convivial Computing at the Forbes College Computing Center .... IV 60

Crystal ....III 72

Designing Local Area Networks for the Academic Environment ....IV 65

Development of a Computerized Practice Simulation ....III 77

Development of an Expert System for CAI in Process Control ....V 104

Dynamic Models in Biochemistry: Using Spreadsheets ....III 81

Educational Software for Structural Engineering .... V 141

Enhancing, Analytical Comp'nt, Quantitative Methods: Statistics .... I 23

Evaluation of a Bayesian Forecasting Program ....III 85

Innovations in Higher Education: A Multimedia Device ....IV 73

Intelligent CIA ....III 93

Interactive Graphics in CAD/CAI in Chemical Engineering ....V 155

Interactive Graphics/Display & Analysis/Earthquake Seismograms .... III 109

Interactive Microcomputer Graphics Software/Modeling Phenomena .... V 168

Interactive Videodisk and Foreign Language Learning .... II 48

IRIS: A Tool for Learning Data Analysis Strategies .... III 114

Laboratories for Introductory Psychology .... I 28

MCADA: Mechanical Computer Aided Dynamic Analysis ....V 179

Measuring Speed of Thought: Microcomputer Psychology Laboratory .... I 34

Molecular Structure Graphics for Undergrad Chemistry Ed. ....III 120

Musical Acoustics Demonstrations/Lessons .... II 60

INDEX BY TITLE, VOLUMES I–V

	Vol	Pg
Networking to Support Education Objectives	IV	76
New Elementary Physics Laboratory Experiments	III	124
PDS–A General Authoring System for Development of Simulations	IV	83
Personal Computers, Economics Curriculum, Yale University	I	40
Philosophical Simulations and Experiments: Scientific Reasoning	II	64
POPGEN: Interactive Simulation Programs, Population Genetics	I	45
Reconstructing A Historical Community: Texas, 1846–1910	I	52
Remote Electronic Music Synthesis Workstations	II	78
Report on Brown/CMU/MIT Window Manager/Environment Convergence	IV	92
Report on Logic: A Workbook	II	83
Representation of Density Distributions on a Sphere	III	131
SAMP – A Program to Compute Sample Size	I	62
Scriptwriter II: From Vaporware to Freeware	IV	98
Software for Printing South Asian Scripts	II	91
SPA The Simulated Political Arena	I	66
Teaching Numerical Math Using Interactive Computer Graphics	V	191
The Cairo Geniza Project	II	101
The Challenge of Qualitative Evaluation	IV	102
The Great American History Machine	I	97
The IBM PC Network: An Introduction to Managing It Wisely	IV	111
The Ins and Outs of Macroeconomic Forecasting: A Tutorial	I	108
The Use of Expert Systems to Teach Veterinary Decision–Making	III	142
The Use of Interactive Thermal Finite Elements	V	196
Tools for Learning at the Interactive Computer Graphics Lab.	IV	152
Tools for Learning Data Analysis	I	85

INDEX BY TITLE, VOLUMES I–V

Vol Pg

Towards an Integrated Design System for the IBM AT .... IV 156

Treaties, Computers, and Undergraduates .... I 112

Using CAMPS to Teach Medical Problem Solving .... III 147

VEGA: A Geometric Modelling System .... V 202

Which Wave Theory .... V 215

Workstations for Computer Music at Carnegie–Mellon .... II 109

You Can Get There From Here .... IV 170

INDEX BY TITLE KEY WORD

Key Word	Vol	Pg
3D	III	2
Academic	IV	65
Acoustics	II	60
AEP	IV	18
	IV	23
Aided	III	16
	V	73
	V	91
	V	179
American	I	97
Analysis	I	23
	I	85
	II	45
	III	109
	III	114
	V	91
	V	179
Architecture	II	11
Arena	I	66
Assisted	I	2
Augmentation	IV	38
Authoring	I	119
	II	32
	II	118
	III	158
	IV	11
	IV	83
	IV	202
	V	231
Bayesian	III	85
Bilingual	II	36
Biochemistry	III	81

INDEX BY TITLE KEY WORD

Key Word	Vol	Pg
CAD	V	155
CAI	V	104
	V	155
Calculus	III	10
CAMPS	III	147
Chemistry	III	2
	III	120
	V	155
Circuits	V	73
Community	I	52
Compute	I	2
	I	13
	I	34
	I	40
	I	62
	I	112
	II	2
	II	45
	II	109
	III	16
	III	29
	III	41
	III	65
	III	77
	IV	38
	IV	54
	IV	60
	IV	152
	V	2
	V	25
	V	73
	V	79
	V	91
	V	95
	V	168
	V	179
	V	191

INDEX BY TITLE KEY WORD

Key Word	Vol	Pg
Conservation	V	2
Constructing	I	52
Control	V	57
	V	104
Convergence	IV	92
Coordinates	III	2
Counseling	I	13
Course	III	29
Crystal	III	72
Curriculum	I	40
Data	I	85
	II	45
	III	114
Database	II	45
Decision	III	142
Decisions	III	142
Demonstration	II	60
Density	III	131
Design	II	11
	IV	65
	IV	156
	V	2
	V	16
	V	36
	V	73
	V	91
Development	III	77
	IV	83
	V	104
Device	IV	73

INDEX BY TITLE KEY WORD

Key Word	Vol	Pg
Dictionary	II	2
Digitizer	IV	11
Display	III	109
Distribution	III	131
Dynamic	III	81
	V	57
	V	95
	V	179
Earth	III	109
Economics	I	40
	I	108
Education	IV	73
	IV	76
	V	16
	V	141
Elementary	III	124
Energy	V	2
Engineering	V	95
	V	141
	V	155
Environment	IV	29
	IV	65
	IV	92
Evaluate	III	85
	IV	18
	IV	102
	III	85
	IV	18
	IV	102
Evolution	III	65
Experiment	II	64
	III	124

INDEX BY TITLE KEY WORD

Key Word	Vol	Pg
Expert	III	142
	V	104
Facility	V	25
Finite	V	196
Forecasting	I	108
	III	85
Foreign	II	32
	II	48
Genetics	I	45
Geniza	II	101
Geometric	V	202
Graphics	III	29
	III	41
	III	109
	III	120
	IV	152
	V	79
	V	155
	V	168
	V	191
Great	I	97
History	I	52
	I	97
Image	V	25
Innovations	IV	73
Instruction	I	13
	III	16
	V	25
	V	95
Integrated	IV	29
	IV	156
	V	73

INDEX BY TITLE KEY WORD

Key Word	Vol	Pg
Interactive	I	45
	II	48
	III	109
	IV	152
	V	36
	V	155
	V	168
	V	191
	V	196
Introduce	I	28
	IV	111
Issues	IV	54
Laboratory	I	28
	I	34
	III	124
	IV	2
Language	II	11
	II	32
	II	48
Learning	I	85
	II	32
	II	36
	II	48
	III	114
	IV	152
Lectures	I	2
Lessons	II	60
Logic	II	45
	II	83
Loop	V	57
Machine	I	97
Macro	I	108

INDEX BY TITLE KEY WORD

Key Word	Vol	Pg
Management	II	45
	IV	54
	IV	92
MCADA	V	179
Mechanical	V	179
Medical	III	147
Methods	I	23
Micro	I	34
	V	2
	V	73
	V	168
Modeling	III	81
	V	168
	V	202
Molecular	III	2
	III	120
Morphology	III	65
Multimedia	IV	73
Music	II	45
	II	60
	II	78
	II	109
Network	IV	65
	IV	76
	IV	111
Neuroanatomy	III	16
Numerical	V	191
Objectives	IV	76
Optics	III	29
Optimal	V	36

INDEX BY TITLE KEY WORD

Key Word **Vol** **Pg**

Philosophy .... II 64

Physics .... III 124

Plant .... III 65

Political .... I 66

Population .... I 45

Practice .... III 77

Printing .... II 91

Problem .... III 147

Process .... II 36
V 25
V 104

Project .... II 101

Psychology .... I 13
I 28
I 34

Quantitative .... I 23

Relativity .... III 41

SAMP .... I 62

Sample .... I 62

Scripts .... II 91

Scriptwriter .... IV 98

Seismic .... III 109

Simulation .... I 45
I 66
II 64
III 65
III 77
IV 38
IV 83

INDEX BY TITLE KEY WORD

Key Word	Vol	Pg
Size	I	62
Social	IV	54
Software	II	91
	III	10
	V	141
	V	168
Solving	III	147
South Asian	II	91
Speed	I	34
Sphere	III	131
Stencil	III	2
Structural	III	120
	V	141
Study	II	45
Support	IV	76
Synthetic	II	78
System	II	32
	III	142
	IV	11
	IV	83
	IV	156
	V	16
	V	36
	V	91
	V	104
	V	202
Teaching	III	142
	III	147
	V	191
Testing	V	57
Text	II	45

INDEX BY TITLE KEY WORD

Key Word	Vol	Pg
Theory	V	215
	V	215
Thermal	V	196
Thermodynamics	V	95
Thought	I	34
Tools	II	32
	II	36
	III	114
	V	2
Training	IV	38
Tutorial	I	108
Undergraduate	I	112
	III	120
Utilities	III	41
Veterinary	III	142
Video	II	48
	IV	2
	IV	11
Videodisk	II	48
Videotex	IV	2
Vision	V	25
Visual	V	79
VLSI	V	16
Wave	V	73
	V	215
Window	IV	92
Word	II	36
Workbook	II	83

Workstations .... II 78
II 109
IV 29

INDEX BY UNIVERSITY, VOLUMES I–V

University	Vol	Pg
Brown University	IV	92
California Institute of Technology	V	73
	V	104
	V	155
Carnegie–Mellon University	I	97
	II	109
	III	10
	III	29
	IV	29
	IV	156
	V	141
	V	202
Columbia University	IV	23
	IV	73
	V	36
Cornell University	II	2
	II	11
	II	32
	II	36
	III	65
	III	93
	V	25
	V	79
	V	168
Massachusetts Institute of Technology	III	41
	III	120
Princeton University	II	101
	IV	60
	IV	152
	V	91
	V	191
University of California, Berkeley	I	2
	III	109
	III	131
	V	57
	V	215

INDEX BY UNIVERSITY, VOLUMES I–V

University	Vol	Pg
University of California, Los Angeles	I	62
	I	108
	II	83
	III	81
	IV	76
	V	2
University of Illinois	II	45
	II	60
	III	124
	III	142
	IV	38
	V	179
University of Maryland	III	16
	IV	83
University of Pennsylvania	I	28
	I	85
	III	147
	IV	11
	IV	54
	IV	65
University of Southern California	III	2
	IV	98
University of Texas	I	23
	I	34
	I	52
	II	64
	II	78
	III	85
	IV	102
University of Washington	I	112
	III	72
	III	114
	IV	2
	IV	18
	IV	111
	V	16
	V	196

INDEX BY UNIVERSITY, VOLUMES I–V

University	Vol	Pg
University of Wisconsin	I	13
	II	91
	III	77
	IV	170
	V	95
Yale University	I	40
	I	45
	I	66
	II	48

TOTAL AUTHOR LIST
IBM ACADEMIC INFORMATION SYSTEMS
472 Wheelers Farms Road
Milford, CT 06460
(203) 783-7226

Mr. Kirk D. Alexander
Princeton University
E-420, Engineering Quadrangle
Princeton, NJ 08544
(609) 452-5420

Mr. James Beauchamp
University of Illinois
5004 Music Building
Urbana, IL 61801
(217) 333-1089

Dr. Donald A. Becker
University of Wisconsin
Department of German
818 Van Hise Hall
Madison, WI 53713
(608) 262-0647

Mr. Stuart Bogom
University of Pennsylvania
Population Studies Center
3718 Locust Walk
Philadelphia, PA 19104-6298
(215) 898-6713

Dr. Michael B. Bolger
University of Southern California
School of Pharmacy
1985 Zonal Avenue PSC-700
Los Angeles, CA 90033
(213) 224-7111

Dr. John E. Bowes
University of Washington
School of Communications, DS-40
Seattle, WA 98195
(206) 543-7998

Mr. Michael J. Braca
Brown University
Box 1946
Brown Station
Providence, RI 02912
(401) 863-3665

Dr. Steven E. Brauth
University of Maryland
Department of Psychology
Room 114SB
Zoo/Psych Building
College Park, MD 20742
(301) 454-6375

Dr. Robert L. Causey
University of Texas at Austin
Department of Philosophy
Austin, TX 78712
(512) 471-4857

Dr. Lynn D. Churchill
University of Washington
Center for Instructional Development
and Research
107 Parrington, DC-07
Seattle, WA 98195
(206) 543-0751

Dr. Richard C. Coddington
University of Illinois
360N Agricultural Engineering
Sciences Building
1304 West Pennsylvania
Urbana, IL 61801
(217) 333-8355

Professor Mark R. Cohen
Princeton University
Jones Hall
Princeton, NJ 08544
(609) 452-5488

Ms. Michelle A. Comeaux
University of Wisconsin
M.A.C.C. Room 1113
1210 W. Dayton Street
Madison, WI 53706
(608) 262-5667

Mr. Richard C. Compton
California Institute of Technology
116-81
Pasadena, CA 91125
(818) 356-4816

Dr. Wayne A. Danielson
University of Texas at Austin
139 Taylor Hall
Project Quest
Austin, TX 78712
(512) 471-4900

Dr. Roger B. Dannenberg
Carnegie Mellon University
Center for Art and Technology
Schenely Park
Pittsburgh, PA 15213
(412) 268-3827

Dr. Bradley W. Dickinson
Princeton University
Department of Electrical Engineering
B316 Engineering Quadrangle
Princeton, NJ 08544
(609) 452-4644

Dr. Ashley F. Emery
University of Washington
Department of Mechanical Engineering
FU-10
Seattle, WA 98195
(206) 543-5338

Dr. Martin Gardy
Cornell University
Medical School
1300 York Avenue
New York, NY 10021
(212) 472-6938

Dr. Clifford B. Gillman
University of Wisconsin
Academic Computing Center
1210 West Dayton Street
Madison, WI 53706
(608) 262-3095

Dr. Paul J. Godley
University of Texas at Austin
College of Pharmacy
Room 5.110
Austin, TX 78712
(512) 471-4048

Dr. Susan E. Godley
University of Texas at Austin
College of Pharmacy
Austin, TX 78712
(512) 471-4048

Dr. Hellmut Golde
University of Washington
301 Administration Building
Mail Stop AE-14
Seattle, WA 98195
(206) 543-0070

Professor Steven M. Goldman
University of California, Berkeley
Department of Economics
Berkeley, CA 94720
(415) 642-2787

Mr. Mark A. Guenin
University of Pennsylvania
School of Medicine
Box 49
Philadelphia, PA 19104
(215) 596-9444

Professor Myron P. Gutmann
University of Texas at Austin
Department of History
Austin, TX 78712
(512) 471-5236

Dr. Wilbur D. Hagemen
Cornell University Medical College
1300 York Avenue
New York, NY 10021
(212) 472-8189

Dr. John Haltiwanger
University of California-Los Angeles
Dept. of Economics
2263 Bunche Hall
Los Angeles, CA 90024
(213) 825-6619

Mr. Roland E. Heersink
California Institute of Technology
M/S 206-41
1201 E. California Boulevard
Pasadena, CA 91125
(818) 356-4869

Ms. Patricia B. Helm
University of Pennsylvania
3333 Walnut
Philadelphia, PA 19104-6193
(215) 898-5237

Dr. John W. Hill
University of Illinois
2136 Music Building
1114 W. Nevada
Urbana, IL 61801
(217) 333-0371

Dr. David A. Hoeltzel
Columbia University
Department of Mechanical Engineering
520 West 120th Street
New York, NY 10027
(212) 280-8372

Mr. Randy D. Holmes
Yale University
c/o D. O. Holmes
17425 Burma
Encino, CA 91316
(203) 785-2654

Dr. Ellis Horowitz
University of Southern California
School of Engineering
Department of Computer Science
Sal 218 MC 0782, University Park
Los Angeles, CA 90089-0782
(213) 743-6453

Dr. Kenneth K. Kidd
Yale University
School of Medicine
Department of Human Genetics
Yale Station Box 3333
New Haven, CT 06520
(203) 785-2654

Dr. Sandy A. Klein
University of Wisconsin
Mechanical Engineering Department
Madison, WI 53706
(608) 262-5972

Mr. Bern E. Kohler
Massachusetts Institute of Technology
Room 2-069
Cambridge, MA 02139
(617) 253-1956

Dr. Eugene A. Kroch
University of Pennsylvania
362 McNeil Building/CR
Department of Economics
Philadelphia, PA 19104
(215) 898-1875

Dr. Peter A. Lachenbruch
University of California-Los Angeles
Division of Biostatistics
UCLA School of Public Health
Los Angeles, CA 90024
(213) 825-5250

Dr. Sandra I. Lamb
University of California-Los Angeles
Department of Cheminstry & Biochemistry
3060 Young Hall
405 Hilgard Avenue
Los Angeles, CA 90024
(213) 825-1853

Dr. James L. Lee
University of Wisconsin
321 Education Building
Madison, WI 53706
(608) 262-8893

Dr. Daniel R. Lewin
California Institute of Technology
Chemical Engineering 206-41
Pasadena, CA 91125
(818) 356-3692

Mr. Anthony J. Lomax
University of California-Berkeley
Seismographic Station
475 Earth Sciences Building
Berkeley, CA 94720
(415) 642-3977

Professor Daniel P. Loucks
Cornell University
Hollister Hall
Ithaca, NY 14853
(607) 255-4896

Mr. Arch B. MacKenzie
Cornell University
Department of Architecture
Sibley Hall
Ithaca, NY 14853
(607) 255-5236

Professor Michael S. Mahoney
Princeton University
220 Palmer Hall
Princeton, NJ 08544
(609) 452-4157

Professor Robert L. Martin
University of California-Los Angeles
Department of Philosophy
Los Angeles, CA 90024
(213) 825-4641

Ms. Deborah McGraw
Cornell University
303 Morrill Hall
Department of Modern Languages &
Linguistics
Ithaca, NY 14853
(607) 255-8289

Dr. Kevin F. Miller
University of Texas at Austin
Department of Psychology
Mezes 330
Austin, TX 78712
(512) 471-4657

Dr. David W. Miller
Carnegie Mellon University
Department of History
Schenley Park
Pittsburgh, PA 15213
(412) 268-2880

Professor Murray A. Milne
University of California-Los Angeles
Graduate School of Architecture
and Urban Planning
405 Hilgard Avenue
Los Angeles, CA 90024
(213) 825-7370

Dr. Norman N. Mintz
Columbia University
205 Low Memorial Library
New York, NY 10027
(212) 280-2488

Dr. John H. Moore
National Science Foundation
1800 G Street, N.W.
Room 518
Washington, DC 20550

Dr. Manfred Morari
California Institute of Technology
Chemical Engineering 206-41
1201 E. California Boulevard
Pasadena, CA 91125
(818) 356-4186

Mr. Leonard Morse-Fortier
Princeton University
Room E-226
Engineering Quadrangle
Princeton, NJ 08544
(609) 452-4666

Mr. Edward J. Mulligan
University of Washington
Department of Geological Sciences, AJ-20
Seattle, WA 98195
(206) 545-2248

Dr. Anthony J. Niesz
Yale University
Yale Language Laboratory
111 Grove Street
New Haven, CT 06511
(203) 436-1746

Dr. Karl J. Niklas
Cornell University
214 Plant Science Building
Ithaca, NY 14850
(607) 255-8409

Dr. James S. Noblitt
Cornell University
Morrill Hall
Department of Modern Languages
and Linguistics
Ithaca, NY 14853
(607) 255-0720

Professor William L. Olbricht
Cornell University
School of Chemical Engineering
Olin Hall
Ithaca, NY 14853
(607) 255-4362

Dr. Irving J. Oppenheim
Carnegie Mellon University
Department of Civil Engineering
Pittsburgh, PA 15213
(412) 268-2950

Professor David I. Owen
Cornell University
Department of Near Eastern Studies
Rockefeller Hall 360
Ithaca, NY 14853
(607) 256-6275

Dr. Slava Paperno
Cornell University
303 Morrill Hall
Ithaca, NY 14853
(607) 255-2322

Professor George F. Pinder
Princeton University
Department of Civil Engineering
E226 Engineering Quadrangle
Princeton, NJ 08544
(609) 452-4602

Dr. Russell Pinkston
University of Texas at Austin
1208 Palo Duro Road
Austin, TX 78757
(512) 471-7764

Dr. Michael E. Pitterle
University of Wisconsin
School of Pharmacy
425 N. Charter Street
Madison, WI 53706
(608) 262-9477

Professor Anthony P. Reeves
Cornell University
School of Electrical Engineering
413 Phillips Hall
Ithaca, NY 14853
(607) 255-4296

Dr. Bruce H. Robinson
University of Washington
Department of Chemistry, BG-10
Seattle, WA 98195
(206) 543-1773

Dr. Peter H. Rohn
University of Washington
Department of Political Science, DO-30
Seattle, WA 98195
(206) 543-8030

Professor Steven J. Rosenstone
Yale University
Political Science Department
3532 Yale Station
New Haven, CT 06520
(203) 436-8529

Dr. David Rutledge
California Institute of Technology
MS 116-81
Pasadena, CA 91125
(818) 356-4806

Dr. Gerhard Schmitt
Carnegie Mellon University
Department of Agriculture
Pittsburgh, PA 15213
(412) 268-3528

Dr. M. William Schwartz
University of Pennsylvania
2nd Floor
Children's Hospital of Philadelphia
34th and Civic Center Boulevard
Philadelphia, PA 19104
(215) 596-9443

Mr. Matthew D. Shapiro
Yale University
Cowles Foundation
2125 Yale Station
New Haven, CT 06520-2125
(203) 436-0605

Dr. Bruce A. Sherwood
Carnegie Mellon University
Center for Design of Educational
Computing
Pittsburgh, PA 15213
(412) 268-8530

Mrs. Judith N. Sherwood
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213

Mr. Randy E. Siler
University of Washington
Center for Instructional Development
and Research
107 Parrington/DC-07
Seattle, WA 98195
(206) 543-6588

Mr. Anthony Skjellum
California Institute of Technology
Mail Stop 206-41
1201 East California Boulevard
Pasadena, CA 91125
(818) 356-4634

Dr. Ronald D. Smith
University of Illinois
College of Veterinary Medicine
2001 S. Lincoln
Urbana, IL 61801
(217) 333-3290

Mr. James H. Smith
University of California-Los Angeles
Graduate School of Management
405 Hilgard Avenue
Los Angeles, CA 90024
(213) 825-2200

Dr. Rodney J. Sobey
University of California-Berkeley
Department of Civil Engineering
412 O'Brien Hall
Berkeley, CA 94720
(415) 642-3162

Dr. Mani Soma
University of Washington
Department of Electrical Engineering
FT-10
Seattle, WA 98195
(206) 545-3810

Mr. Jorge B. Sousa
University of California-Berkeley
Department of Civil Engineering
106 McLaughlin Hall
Berkeley, CA 94720
(415) 231-9560

Dr. J. I. Steinfeld
Massachusetts Institute of Technology
Room 2-221 MIT
Cambridge, MA 02139
(617) 253-4525

Dr. Michael K. Stenstrom
University of California-Los Angeles
4532 Boelter Hall
Los Angeles, CA 90024
(213) 825-1408

Mr. Lars P. Stixrude
University of California-Berkeley
Department of Geology
Berkeley, CA 94720
(415) 642-9075

Dr. Chandler Stolp
University of Texas at Austin
LBJ School of Public Affairs
Austin, TX 78713
(512) 471-4962

Dr. Edwin F. Taylor
Massachusetts Institute of Technology
Room E15-232
Cambridge, MA 02139
(617) 253-7433

Mr. Bruce Tetelman
Columbia University
612 West 115th Street
Department of Computer Science
New York, NY 10025
(212) 280-3758

Dr. Andries van Dam
Brown University
Computer Science Department
Box 1910
Brown Station
Providence, RI 02912
(401) 863-3300

Dr. Ned S. VanderVen
Carnegie Mellon University
Department of Physics
Pittsburgh, PA 15213
(412) 268-2766

Dr. Russell C. Walker
Carnegie Mellon University
Department of Mathematics
Pittsburgh, PA 15213
(412) 268-2546

Dr. Martha H. Weller
University of Illinois
Institute of Aviation
Q5, Aviation Research Laboratory
#1 Airport Road
Savoy, IL 61874
(217) 333-3162

Mr. Christopher D. Wickens
University of Illinois
836 Psychology Building
Urbana, IL 61801
(217) 333-3162

Dr. David R. Williams
University of Pennslyvania
Department of Psychology
3815 Walnut Street
Philadelphia, PA 19104-6196
(215) 898-6947

Mr. Wyman Williams
California Institute of Technology
1201 East California Boulevard
Pasadena, CA 91125
(818) 356-6811

Dr. Mark Winer
University of Maryland
206 Dale Drive
Silver Spring, MD 20910
(301) 395-3087

Mr. Robert F. Woodbury
Carnegie Mellon University
Pittsburgh, PA 15213
(412) 268-8853

Mr. Weiguang Zhang
Carnegie Mellon University
PH103
Pittsburgh, PA 15213
(412) 268-3777